A project to predict whether a tech employee may require mental health treatment, based on background data acquired through a global survey.

Mental Health in the Tech Industry

Module Title: CMP7247 – Artificial Intelligence Fundamentals

School: School of Computing and Digital Technology

Module Co-ordinator: Dr Debashish Das

Lindelani Moyo (S18125918)

Submitted on 10/05/22

Abstract

Mental health disorders are a global issue that affects over 300 million people. In recent years, there has been a growing recognition of the impact that the workplace has on an employee's mental health. Using the largest mental health in tech survey dataset available today, compiled by 1259 tech employees, this project will use machine learning methods to select and rank the most important factors that affect the mental health of tech employees, and will build a prediction model based on supervised learning algorithms (including logistic regression, K nearest neighbour, decision tree classifier, and random forest) and the machine learning algorithm: neural networks.

Key Words

Artificial Intelligence, Mental Health, Tech Industry, Logistic Regression, Labelled Data, Supervised Data, Survey Analysis, Data Analysis, Data Visualization, Decision Trees, Random Forest, Machine Learning.

Contents Page

Contents **Page**

Introduction

According to the World Health Organization's (WHO) Constitution (2018), “Health is a state of complete physical, mental and social well-being and not merely the absence of disease or infirmity". This definition clarifies that mental health comprises far more than the absence of mental conditions or disabilities. Thus signifying that mental health is an important and integral aspect of overall health and wellness.

Research from the study ‘Mental Health’ (Dattani, Ritchie and Roser, 2018) which analysed ‘Global Burden of Disease’ (GBD, 2016) data estimated that, 792 million individuals worldwide suffer from some form of mental health disorder, accounting for 10.7% of the global population. There are many different types of mental health disorders, and each has its own unique set of symptoms and characteristics; the two most widespread “were anxiety (284 million people, 3.8% of the population) and depression (264 million people, 3.4% of the population)” (Dattani, Ritchie and Roser, 2018). While diagnosing and treating mental health disorders can be challenging, many conditions can be efficiently treated at a minimal cost (taking into account access to public health care). However, the gap between those in need of care and those who have access to care continues to be vast.

Background

In recent years, there has been a rising acknowledgment of the importance of supporting employees with mental health issues in the workplace. According to recent statistics released by the British Interactive Media Association (BIMA), “mental health in tech is currently in a poor state, and some would even go as far as saying its reaching crisis point” (BIMA, 2019). However, this seems ironic for an industry that "is booming" (FDM, 2016) with high salaries, luxurious benefits and shows no signs of slowing down, having recently been reported to be growing six times faster than any other industry (Jack, 2020) with over “1000% growth between 2010 and 2020” according to the British Government's Digital Economy Council (Dunne and Hewson, 2021).

Prior to 2014, mental health disorders in the technology industry were underreported and understudied. Many publications, use mental health prediction as an important aspect in reducing the time it takes to recognise the risks and factors of an individual developing significant mental health disorders, according to past mental health statistical data obtained from a wide range of industries such as health and social care. Mental health prediction can give a theoretical foundation for public health departments and internal corporate HR departments to develop psychological support programmes to help their employees who suffer from mental health disorders.

Aims and Objectives

Aims

Using supervised learning algorithms, this project attempts to build, evaluate and create a model based on the survey dataset which can predict whether or not a tech employee may require mental health treatment.

Objectives

The main objectives of this project:

- Implementing data pre-processing, visualisation, feature selection.
- Evaluate and compare the effectiveness of the built models.
- Critically discuss the AI project life cycle compliance with the AI ethics.

Dataset

This raw dataset is a survey which has been uploaded on Kaggle by Open Sourcing Mental Illness (OSMI). OSMI is a non-profit corporation, founded by developer Ed Finkler in 2013. OSMI's focus is on raising awareness and educating mental health within the tech industry (Chan, 2020).

Although OSMI publish a new survey onto Kaggle every other year to see, the raw dataset used in this investigation was published in 2014 (https://www.kaggle.com/osmi/mental-health-in-tech-survey). This specific dataset was used as the more recent datasets are missing too many values which at first glance could be a risk due to their relevance in gaining significant findings and forming a suitable conclusion at the end of this report. Furthermore, the 2014 dataset was generated from "the largest survey done on mental health in the tech industry" at its time (OSMH, 2014) (as shown in **Appendices 1**).

The following section presents the information in the dataset:

```
import pandas as pd

#mh = mental health

techmhData=pd.read_csv('survey.csv')
```

Figure 1. *Reading the data from csv file.*

What are the features of this dataset?

In Figure 1, pandas is imported and the survey dataset is accessed and read. The features of each column are then printed and shown, below in Figure 2.

```
#printing the column names
print(techmhData.columns)
```

```
Index(['Timestamp', 'Age', 'Gender', 'Country', 'state', 'self_employed',
       'family_history', 'treatment', 'work_interfere', 'no_employees',
       'remote_work', 'tech_company', 'benefits', 'care_options',
       'wellness_program', 'seek_help', 'anonymity', 'leave',
       'mental_health_consequence', 'phys_health_consequence', 'coworkers',
       'supervisor', 'mental_health_interview', 'phys_health_interview',
       'mental_vs_physical', 'obs_consequence', 'comments'],
      dtype='object')
```

Figure 2. *Mental health features in the survey dataset.*

What are the first five rows of this dataset?

In Figure 3, the head() function is used to quickly test whether the dataset has the right types of data in it. The results are returned and shown below.

```
#getting the first five rows of the df
techmhData.head()
```

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_health_consequ
0	2014-08-27 11:29:31	37	Female	United States	IL	NaN	No	Yes	Often	6-25	...	Somewhat easy	
1	2014-08-27 11:29:37	44	M	United States	IN	NaN	No	No	Rarely	More than 1000	...	Don't know	M
2	2014-08-27 11:29:44	32	Male	Canada	NaN	NaN	No	No	Rarely	6-25	...	Somewhat difficult	
3	2014-08-27 11:29:46	31	Male	United Kingdom	NaN	NaN	Yes	Yes	Often	26-100	...	Somewhat difficult	
4	2014-08-27 11:30:22	31	Male	United States	TX	NaN	No	No	Never	100-500	...	Don't know	

5 rows × 27 columns

Figure 3. *To find out the objects in the rows of the dataset.*

What is the dataset size?

Figure 4 shows that this raw dataset contains 27 features/columns with 1259 rows and includes responses from employees working within tech companies who suffer from mental health disorders (both that have been professionally diagnosed and undiagnosed).

```
#getting the number of rows and columns of the df
techmhData.shape
```

```
(1259, 27)
```

Figure 4. *Dataset size.*

What are the descriptive statistics of the dataset?

```
#produces descriptive statistics of the dataset
techmhData.describe(include='all')
```

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_healt
count	1259	1.259000e+03	1259	1259	744	1241	1259	1259	995	1259	...	1259	
unique	1246	NaN	49	48	45	2	2	2	4	6	...	5	
top	2014-08-27 12:54:11	NaN	Male	United States	CA	No	No	Yes	Sometimes	6-25	...	Don't know	
freq	2	NaN	615	751	138	1095	767	637	465	290	...	563	
mean	NaN	7.942815e+07	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
std	NaN	2.818299e+09	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
min	NaN	-1.726000e+03	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
25%	NaN	2.700000e+01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
50%	NaN	3.100000e+01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
75%	NaN	3.600000e+01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
max	NaN	1.000000e+11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	

11 rows × 27 columns

Figure 5. *Collective properties and statistical analysis of the elements of the dataset.*

What is the metadata information of the data?

Figure 6, shows information of the metadata such as feature datatypes and has checked for null values, which there are none of.

```
#checking the datatypes of each column
techmhData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 27 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Timestamp                  1259 non-null   object
 1   Age                        1259 non-null   int64 
 2   Gender                     1259 non-null   object
 3   Country                    1259 non-null   object
 4   state                      744 non-null    object
 5   self_employed              1241 non-null   object
 6   family_history             1259 non-null   object
 7   treatment                  1259 non-null   object
 8   work_interfere             995 non-null    object
 9   no_employees               1259 non-null   object
 10  remote_work                1259 non-null   object
 11  tech_company               1259 non-null   object
 12  benefits                   1259 non-null   object
 13  care_options               1259 non-null   object
 14  wellness_program           1259 non-null   object
 15  seek_help                  1259 non-null   object
 16  anonymity                  1259 non-null   object
 17  leave                      1259 non-null   object
 18  mental_health_consequence  1259 non-null   object
 19  phys_health_consequence    1259 non-null   object
 20  coworkers                  1259 non-null   object
 21  supervisor                 1259 non-null   object
 22  mental_health_interview    1259 non-null   object
 23  phys_health_interview      1259 non-null   object
 24  mental_vs_physical         1259 non-null   object
 25  obs_consequence            1259 non-null   object
 26  comments                   164 non-null    object
dtypes: int64(1), object(26)
memory usage: 265.7+ KB
```

Figure 6. *Metadata of the dataset.*

Problem Statement

Recent research on this dataset has focused on the respondents' attitudes towards mental health (Johnson, 2019) (Gupta, 2018), the impact of mental health on their work (Larson, 2021), and the prevalence of mental health disorders in the tech industry (OSMH, 2014). Furthermore, these studies revealed that the majority of respondents did not believe that sharing their mental health issues with their employer would be beneficial to them and could have bad implications, and that there was a need for helpful resources within the workplace. This created the difficulty and challenge of not only how can employees obtain help from their employer if they seek mental health treatment, but also how would they be encouraged to seek mental health therapy if they choose to remain silent.

As a result, it brought attention to the problem that there are a lack of solutions on predicting whether or not a tech employee may require mental health treatment. This could be achieved by building an AI model to calculate this issue and assess whether or not a tech employee may require mental health treatment based on background data acquired from the survey. Moreover, this could also become a tool for internal HR departments to support their employees by determining which support programmes to deliver to those suffering from factors linked to mental disorders, which may end up encouraging them to pursue treatment.

Artificial Intelligence Models

1. Summary of the approach

This study will address the problem statement by developing a project that predicts if a tech employee may need mental health treatment. This will be implemented by selecting specific features to build a predictive classification model, and then testing supervised learning algorithms including: logistic regression, K nearest neighbour, decision tree classifier, and random forest. The machine learning algorithm: neural network will then be applied to improve the prediction accuracy, as recommended by Mesquita (2021), and the strongest model will be trained and taught to make predictions on the test set.

2. Data pre-processing, visualisation, feature selection

Data pre-processing

Importing the necessary dependencies for the project

```
import pandas as pd
import numpy as np
import seaborn as sns

#for visualisation and graphing
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
import warnings
warnings.filterwarnings("ignore")

from scipy import stats
from scipy.stats import randint

#for preprocessing
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import binarize, LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification

#for models
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier

#for validation libraries
from sklearn import metrics
from sklearn.metrics import accuracy_score, mean_squared_error, precision_recall_curve
from sklearn.model_selection import cross_val_score

#for neural network
from sklearn.model_selection import RandomizedSearchCV
```

Figure 7. *Importing the dependencies.*

Cleaning

To begin, the data needs to be cleaned. This process is necessary to improve the data quality, ensuring that the data is consistent and usable increasing overall productivity.

```
#finding the missing values
techmhData.isna()
```

	Timestamp	Age	Gender	Country	state	self_employed	family_history	treatment	work_interfere	no_employees	...	leave	mental_health_consequ
0	False	False	False	False	False	True	False	False	False	False	...	False	
1	False	False	False	False	False	True	False	False	False	False	...	False	
2	False	False	False	False	True	True	False	False	False	False	...	False	
3	False	False	False	False	True	True	False	False	False	False	...	False	
4	False	False	False	False	False	True	False	False	False	False	...	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1254	False	False	False	False	True	False	False	False	True	False	...	False	
1255	False	False	False	False	False	False	False	False	False	False	...	False	
1256	False	False	False	False	False	False	False	False	False	False	...	False	
1257	False	False	False	False	False	False	False	False	True	False	...	False	
1258	False	False	False	False	False	False	False	False	False	False	...	False	

1259 rows × 27 columns

Figure 8. *Finding the missing values.*

Observing the data in Figure 8, there are a few missing values (these can be identified via 'true' and 'false' for no missing value).

It is necessary to infer the missing values. As a result, we must determine the amount of missing values in each column.

```
#get the number of missing values of each column
techmhData.isna().sum()
```

```
Timestamp                      0
Age                            0
Gender                         0
Country                        0
state                        515
self_employed                 18
family_history                 0
treatment                      0
work_interfere               264
no_employees                   0
remote_work                    0
tech_company                   0
benefits                       0
care_options                   0
wellness_program               0
seek_help                      0
anonymity                      0
leave                          0
mental_health_consequence      0
phys_health_consequence        0
coworkers                      0
supervisor                     0
mental_health_interview        0
phys_health_interview          0
mental_vs_physical             0
obs_consequence                0
comments                    1095
dtype: int64
```

Figure 9. *Number of missing values of each column.*

Figure 9 shows that there are 4 columns with missing values.

The mean is calculated over the rows to identify the percentage of missing values in each column, which is used to determine the data's central tendency.

```
#using mean over the rows (to add up all the trues and divide it by the total number in each column, to get the % missing val
techmhData.isna().mean()
```

```
Timestamp                    0.000000
Age                          0.000000
Gender                       0.000000
Country                      0.000000
state                        0.409055
self_employed                0.014297
family_history               0.000000
treatment                    0.000000
work_interfere               0.209690
no_employees                 0.000000
remote_work                  0.000000
tech_company                 0.000000
benefits                     0.000000
care_options                 0.000000
wellness_program             0.000000
seek_help                    0.000000
anonymity                    0.000000
leave                        0.000000
mental_health_consequence    0.000000
phys_health_consequence      0.000000
coworkers                    0.000000
supervisor                   0.000000
mental_health_interview      0.000000
phys_health_interview        0.000000
mental_vs_physical           0.000000
obs_consequence              0.000000
comments                     0.869738
dtype: float64
```

Figure 10. *Mean of missing values of each column.*

As shown in Figure 10, 40% of the values are missing in 'state'. Although 40% is a large percentage, state only applies if the respondents' country of origin is 'US'. However, this column is not useful when building the models. The other column 'comments' is equally insignificant, so it will be dropped, along with 'Timestamp'. Keeping missing values in datasets may skew the results and/or reduce the accuracy of the models once they are built.

'Self_employed' and 'work_interfere' are the final remaining columns which will be filled because they have low numbers and are relevant to the investigation.

```
#Dropping the columns "Timestamp", "comments" and "state" ("self_employed will be filled as its a low number")
techmhData = techmhData.drop(['comments'], axis= 1)
techmhData = techmhData.drop(['state'], axis= 1)
techmhData = techmhData.drop(['Timestamp'], axis= 1)
```

Figure 11. *Dropping the comments, state and timestamp.*

Now we check for missing values to see if the 3 columns have been dropped successfully. 264 values remain, these are from the 'work_interfere' column.

```
#checking for missing values
techmhData.isnull().sum().max()
```

```
264
```

Figure 12. *Checking for remaining missing values.*

Now we inspect at the dataframe's head to view the rows that have been changed to confirm that the three columns have been removed.

```
#checking the amended rows
techmhData.head()
```

	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	no_employees	remote_work	tech_company	...	anonymity	leave
0	37	Female	United States	NaN	No	Yes	Often	6-25	No	Yes	...	Yes	Somewhat easy
1	44	M	United States	NaN	No	No	Rarely	More than 1000	No	No	...	Don't know	Don't know
2	32	Male	Canada	NaN	No	No	Rarely	6-25	No	Yes	...	Don't know	Somewhat difficult
3	31	Male	United Kingdom	NaN	Yes	Yes	Often	26-100	No	Yes	...	No	Somewhat difficult
4	31	Male	United States	NaN	No	No	Never	100-500	Yes	Yes	...	Don't know	Don't know

5 rows × 24 columns

Figure 13. *Checking the amended rows of the dataset.*

Cleaning – NaN (not a number)

As shown in Figure 13, the dataset contains NaN (not a number) to represent missing values. We will convert them into floats.
For each data type, we first assign default values.

```
#assigning default values
defaultInt = 0
defaultString = 'NaN'
defaultFloat = 0.0
```

Figure 14. *Assigning default values.*

The data types of each column are then organised into lists.

```
#converting the specified value
intFeatures = ['Age']
stringFeatures = ['Gender', 'Country', 'self_employed', 'family_history', 'treatment', 'work_interfere',
                  'no_employees', 'remote_work', 'tech_company', 'anonymity', 'leave', 'mental_health_consequence',
                  'phys_health_consequence', 'coworkers', 'supervisor', 'mental_health_interview', 'phys_health_interview',
                  'mental_vs_physical', 'obs_consequence', 'benefits', 'care_options', 'wellness_program',
                  'seek_help']
floatFeatures = []
```

Figure 15. *Converting the specified values.*

The NaN's are then cleaned by replacing them with the default value of 0.

```
#cleaning NaN
for feature in techmhData:
    if feature in intFeatures:
        techmhData[feature] = techmhData[feature].fillna(defaultInt)
    elif feature in stringFeatures:
        techmhData[feature] = techmhData[feature].fillna(defaultString)
    elif feature in floatFeatures:
        techmhData[feature] = techmhData[feature].fillna(defaultFloat)
    else:
        print('Error : Feature %s not recognized.' % feature)

techmhData.head()
```

Figure 16. *Cleaning NaN values.*

Cleaning – ‘Gender’ column

To clean the gender column we begin by lower casing all of the elements as they are strings.

```
#lower case all elements
gender = techmhData['Gender'].str.lower()
```

Figure 17. *Lower casing the elements in the ‘gender’ column.*

To create a list of unique values present in the column.

```
#selecting unique elements
gender = techmhData['Gender'].unique()
```

Figure 18. *Selecting the unique elements.*

The genders are then grouped into three sets, ‘male’, ‘female’, ‘trans’ (for any other gender not corresponding to their birth sex (LGBTQ+ identifier)).

```
#different gender groupings
male_str = ["male", "m", "male-ish", "maile", "mal", "male (cis)", "make", "male ", "man","msle", "mail", "malr","cis man", "

female_str = ["cis female", "f", "female", "woman",  "femake", "female ","cis-female/femme", "female (cis)", "femail"]

trans_str = ["trans-female", "something kinda male?", "queer/she/they", "non-binary","nah", "all", "enby", "fluid", "genderqu
```

Figure 19. *Grouping the responses with their identified gender.*

Each row is then iterated in the data frame to replace the original object.

```
#iterating each row
for (row, col) in techmhData.iterrows():
    if str.lower(col.Gender) in male_str:
        techmhData['Gender'].replace(to_replace = col.Gender, value = 'male', inplace = True)

    if str.lower(col.Gender) in female_str:
        techmhData['Gender'].replace(to_replace = col.Gender, value = 'female', inplace = True)

    if str.lower(col.Gender) in trans_str:
        techmhData['Gender'].replace(to_replace = col.Gender, value = 'trans', inplace = True)


random_list = ['A little about you', 'p']
techmhData = techmhData[~techmhData['Gender'].isin(random_list)]
```

Figure 20. *Iterating each row.*

Now we check the success of the iterated unique elements. As shown in Figure 21, there are now 3 gender responses: ‘female’, ‘male’ and ‘trans’.

```
#checking for unique elements in Gender column
print(techmhData['Gender'].unique())
```

```
['female' 'male' 'trans']
```

Figure 21. *Checking the unique elements in the ‘gender’ column.*

Cleaning – 'Age' column

Since the data is skewed, we'll need to fill in the missing parts to clean up the age column. Therefore, median imputation will be used.

```
#fill missing elements with mean
techmhData['Age'].fillna(techmhData['Age'].median(), inplace = True)
```

Figure 22. *Preparing the data frame to make the pending changes permanent.*

As 18 was the lowest legal age for respondents to submit a response to the survey and 120 was the greatest age amongst the respondents, the median values filled the middle number between 18 and 120.

```
#fill with median values < 18 and > 120
s = pd.Series(techmhData['Age'])
s[s<18] = techmhData['Age'].median()
techmhData['Age'] = s
s = pd.Series(techmhData['Age'])
s[s>120] = techmhData['Age'].median()
techmhData['Age'] = s
```

Figure 23. *Filling in the missing elements in 'age' column with the median value.*

As we move from a continuous to a categorical variable, we must now segment and sort the data into a new column called 'age_range'.

```
#ranges of age
techmhData['age_range'] = pd.cut(techmhData['Age'], [0,20,30,65,100], labels = ["0-20", "21-30", "31-65", "66-100"], include_
```

Figure 24. *Ranges of age*

Cleaning – 'self employed' column

The 'self employed' column will be cleaned by replacing all occurrences of the old substring with the new substring 'No,' leaving only yes and no responses. The column's unique values are then returned.

```
techmhData['self_employed'] = techmhData['self_employed'].replace([defaultString], 'No')
print(techmhData['self_employed'].unique())
```

```
['No' 'Yes']
```

Figure 25. *Cleaning the 'self_employed' column.*

Cleaning – 'work interfere' column

The process used to clean the 'self_employed' column is then repeated for the column 'work_interfere' but with the 'NA' values (which are visible in survey.csv) as substring 'Don\'t know.'

```
techmhData['work_interfere'] = techmhData['work_interfere'].replace([defaultString], 'Don\'t know')
print(techmhData['work_interfere'].unique())
```

```
['Often' 'Rarely' 'Never' 'Sometimes' "Don't know"]
```

Figure 26. *Checking the 'work_interfere' column.*

Encoding

Since the vast majority of the data contains categorical variables, it will not be suitable for processing, so we must convert it into a format that can program data processing and execution. To achieve this, we implement label encoding, which normalises labels and converts each value (non-numerical labels) in a column to a number (numerical labels).

```
labelDict = {}
for feature in techmhData:
    le = preprocessing.LabelEncoder()
    le.fit(techmhData[feature])
    le_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
    techmhData[feature] = le.transform(techmhData[feature])

#get labels
    labelKey = 'label_' + feature
    labelValue = [*le_name_mapping]
    labelDict[labelKey] =labelValue

for key, value in labelDict.items():
    print(key, value)

techmhData.head()
```

```
label_Age [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 4
6, 47, 48, 49, 50, 51, 53, 54, 55, 56, 57, 58, 60, 61, 62, 65, 72]
label_Gender ['female', 'male', 'trans']
label_Country ['Australia', 'Austria', 'Belgium', 'Bosnia and Herzegovina', 'Brazil', 'Bulgaria', 'Canada', 'China', 'Colomb
ia', 'Costa Rica', 'Croatia', 'Czech Republic', 'Denmark', 'Finland', 'France', 'Georgia', 'Germany', 'Greece', 'Hungary',
'India', 'Ireland', 'Israel', 'Italy', 'Japan', 'Latvia', 'Mexico', 'Moldova', 'Netherlands', 'New Zealand', 'Nigeria', 'Nor
way', 'Philippines', 'Poland', 'Portugal', 'Romania', 'Russia', 'Singapore', 'Slovenia', 'South Africa', 'Spain', 'Sweden',
'Switzerland', 'Thailand', 'United Kingdom', 'United States', 'Uruguay', 'Zimbabwe']
label_self_employed ['No', 'Yes']
label_family_history ['No', 'Yes']
label_treatment ['No', 'Yes']
label_work_interfere ["Don't know", 'Never', 'Often', 'Rarely', 'Sometimes']
label_no_employees ['1-5', '100-500', '26-100', '500-1000', '6-25', 'More than 1000']
label_remote_work ['No', 'Yes']
label_tech_company ['No', 'Yes']
label_benefits ["Don't know", 'No', 'Yes']
label_care_options ['No', 'Not sure', 'Yes']
label_wellness_program ["Don't know", 'No', 'Yes']
label_seek_help ["Don't know", 'No', 'Yes']
label_anonymity ["Don't know", 'No', 'Yes']
label_leave ["Don't know", 'Somewhat difficult', 'Somewhat easy', 'Very difficult', 'Very easy']
```

```
label_mental_health_consequence ['Maybe', 'No', 'Yes']
label_phys_health_consequence ['Maybe', 'No', 'Yes']
label_coworkers ['No', 'Some of them', 'Yes']
label_supervisor ['No', 'Some of them', 'Yes']
label_mental_health_interview ['Maybe', 'No', 'Yes']
label_phys_health_interview ['Maybe', 'No', 'Yes']
label_mental_vs_physical ["Don't know", 'No', 'Yes']
label_obs_consequence ['No', 'Yes']
label_age_range ['0-20', '21-30', '31-65', '66-100']
```

	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	no_employees	remote_work	tech_company	...	leave	mental_health_c
0	19	0	44	0	0	1	2	4	0	1	...	2	
1	26	1	44	0	0	0	3	5	0	0	...	0	
2	14	1	6	0	0	0	3	4	0	1	...	1	
3	13	1	43	0	1	1	2	2	0	1	...	1	
4	13	1	44	0	0	0	1	1	1	1	...	0	

5 rows × 25 columns

Figure 27. *Checking the 'work_interfere' column.*

We do the final check for any missing data and the percentage of missing values in each column.

```
#check for missing data
total = techmhData.isnull().sum().sort_values(ascending = False)
percent = (techmhData.isnull().sum() / techmhData.isnull().count()).sort_values(ascending = False)
missing_techmhData = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_techmhData.head()
print(missing_techmhData)
```

```
                           Total  Percent
age_range                      0      0.0
care_options                   0      0.0
Gender                         0      0.0
Country                        0      0.0
self_employed                  0      0.0
family_history                 0      0.0
treatment                      0      0.0
work_interfere                 0      0.0
no_employees                   0      0.0
remote_work                    0      0.0
tech_company                   0      0.0
benefits                       0      0.0
wellness_program               0      0.0
obs_consequence                0      0.0
seek_help                      0      0.0
anonymity                      0      0.0
leave                          0      0.0
mental_health_consequence      0      0.0
phys_health_consequence        0      0.0
coworkers                      0      0.0
supervisor                     0      0.0
mental_health_interview        0      0.0
phys_health_interview          0      0.0
mental_vs_physical             0      0.0
Age                            0      0.0
```

Figure 28. *Checking for missing data.*

Visualisation

Correlation Matrix

Now we will observe the correlation between the categories of each variable. The correlation matrix will also advise which columns we will need to drop according to the range. From Figure 29 we can see that variables 'treatment' and 'work_interfere' have a strong correlation with each other.

```
#variability comparison between categories of variables using correlation matrix
corrmat = techmhData.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
plt.show()
```

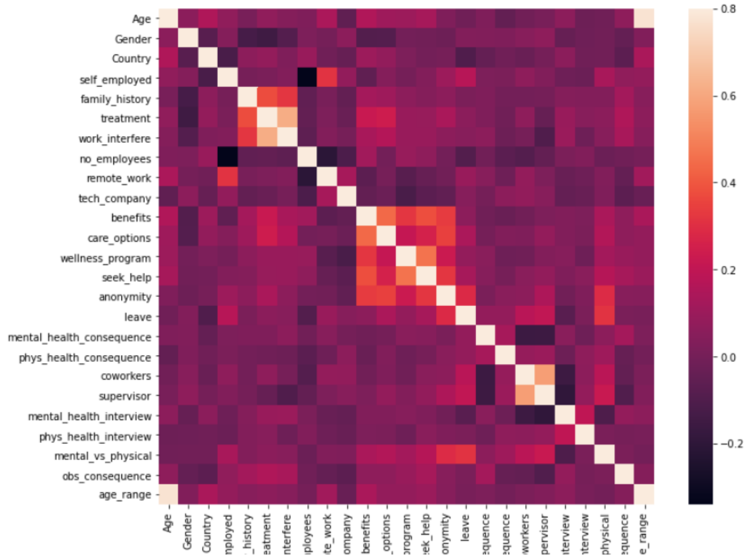


Figure 29. *Correlation Matrix.*

As treatment is our subject of focus, we will target the 'treatment' variable, shown in Figure 30.

```
#"treatment" correlation matrix
k = 10 #number of variables for heatmap
cols = corrmat.nlargest(k, 'treatment')['treatment'].index
cm = np.corrcoef(techmhData[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10}, yticklabels=cols.values, xticklab
plt.show()
```

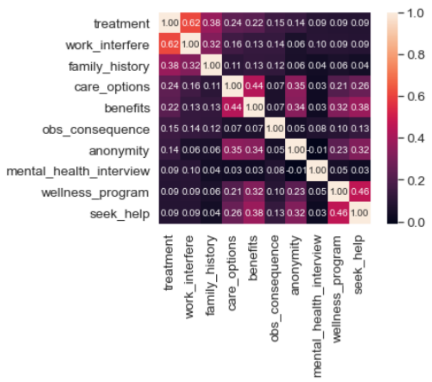


Figure 30. *'Treatment' variable Correlation Matrix.*

Exploratory Data Analysis (EDA)

Through an exploratory data analysis (EDA), we will now visualise the data, summarising key insights and characteristics.

Distribution and Density by "Age" variable

As indicated in Figure 31, employees who complete the survey are in their late twenties to early forties and are likely to be in mid- to senior-level positions. The age distribution is right-skewed, which is to be expected in the tech industry, which has a younger workforce.

```
#distribution and density by "Age"
plt.figure(figsize=(12,8))
sns.distplot(techmhData["Age"], bins=24)
plt.title("Distribution and Density by Age")
plt.xlabel("Age")
```

```
Text(0.5, 0, 'Age')
```

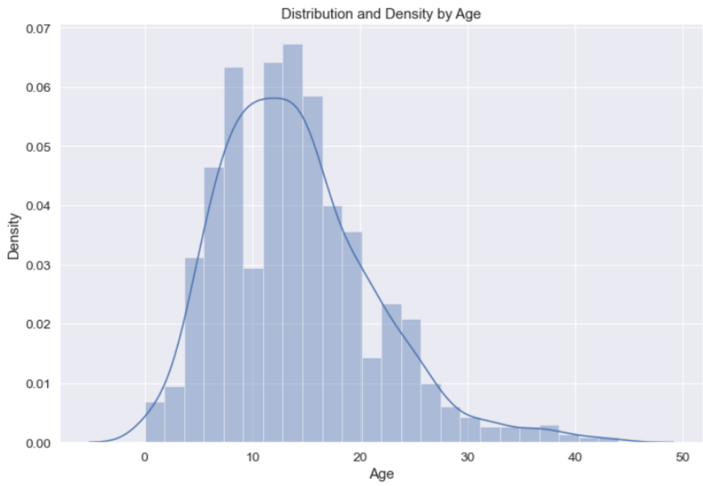


Figure 31. *Distribution and Density by Age.*

'Have you got treatment for a mental health condition?' - Separated by "Treatment" variable

Figure 32 indicates that there is no statistically significant age difference between employees who receive treatment and those who do not.

```
#separate by "treatment" (treated) or not
g = sns.FacetGrid(techmhData, col='treatment', size=5)
g = g.map(sns.distplot, "Age")
```

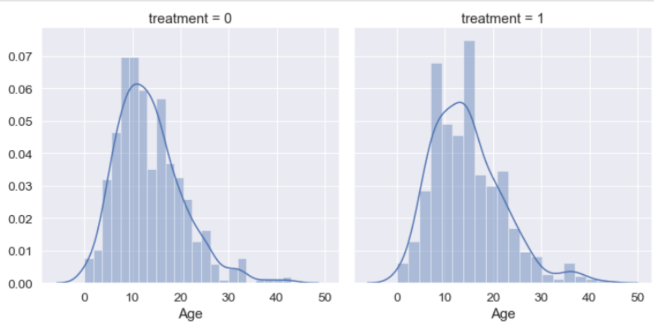


Figure 32. *Separating respondents' response to treated for their mental health condition or not.*

The following findings are divided into two categories to determine what factors motivate employees to seek treatment. Employee profiling and mental health facilities will being explored:

- ***Characterization of employee's***

What is the probability of a Mental Health Disorder using the "Age" and "Gender" variables?

Figure 34, shows that over 80% of transgender employees between the ages of 21 and 30 are likely to have a mental health disorders. Females and transgenders between the ages of 31 and 65 are next, followed by females under the age of 30.

```
#nested barplot to show probabilities for "Age" and "Gender"
o = labelDict['label_age_range']

g = sns.factorplot(x="age_range", y="treatment", hue="Gender", data=techmhData, kind="bar",  ci=None, size=5, aspect=2, leger
g.set_xticklabels(o)

plt.title('Probability of a Mental Health Disorder')
plt.ylabel('Probability x 100')
plt.xlabel('Age')

# replace legend labels
new_labels = labelDict['label_Gender']
for t, l in zip(g._legend.texts, new_labels): t.set_text(l)

#positioning the legend
g.fig.subplots_adjust(top=0.9,right=0.8)

plt.show()
```

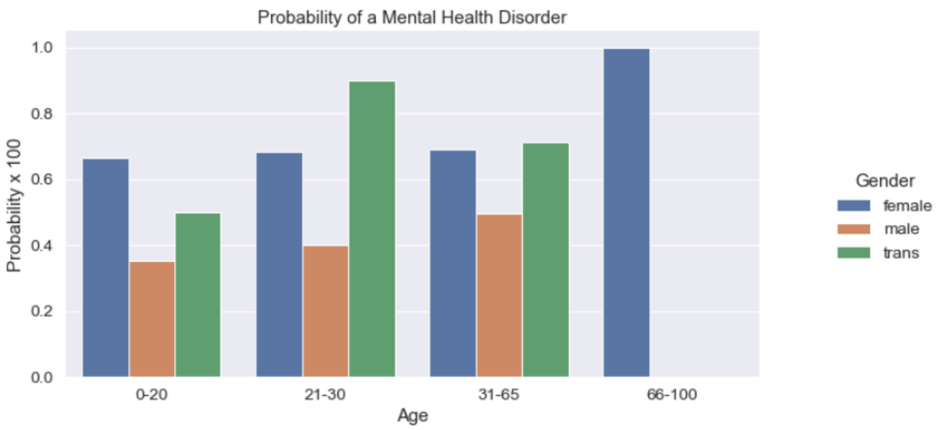


Figure 34. *Probability of a mental health disorder based on surveyee's gender and age.*

(It is important to bear in mind here that the gender **minority groups** in technology are showing the highest risk of mental health disorders).

What is the probability of a Mental Health Disorder using the "Family_History" variable?

Figure 35 reveals that over 80% of transgender employees have a mental health disorder in their family. Female employees are closely following this. Around 30% of males, on the other hand, believe they do not come from a family with a history of mental disorders. With 40% of respondents reporting a family history of mental disorder, the plot indicates that they are more likely to seek treatment than those without a family history. This makes sense because people who are aware of their family history are more likely to be receptive of mental health illnesses. This is because a family history of mental illness is a significant risk factor for many mental health disorders.

```
#barplot to show probabilities for "family_history"
o = labelDict['label_family_history']
g = sns.factorplot(x="family_history", y="treatment", hue="Gender", data=techmhData, kind="bar", ci=None, size=5, aspect=2, l
g.set_xticklabels(o)
plt.title('Probability of a Mental Health Disorder')
plt.ylabel('Probability x 100')
plt.xlabel('Family History')

#replace legend labels
new_labels = labelDict['label_Gender']
for t, l in zip(g._legend.texts, new_labels): t.set_text(l)

#positioning the legend
g.fig.subplots_adjust(top=0.9,right=0.8)

plt.show()
```

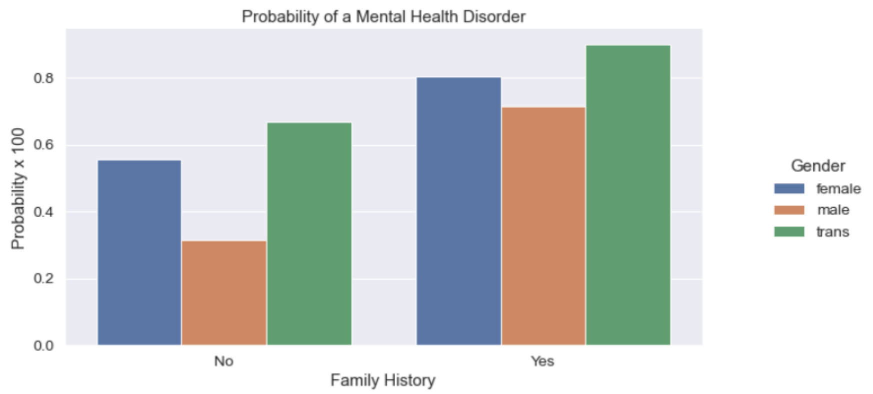


Figure 35. *Employee's probability of having a mental health disorder based on their gender and family history.*

- ***Employees mental health facilities***

"Care_Options" variable – Are employees aware of the mental health care options their employer provides?

Figure 36 indicates that the majority of female employees are aware of their company's mental health 'care options,' which is a belief that is progressively shared by the male employees. However, a large number of transgender employees are unsure about the 'care options' that are available to them.

```
#barplot to show probabilities for "care_options"
o = labelDict['label_care_options']
g = sns.factorplot(x="care_options", y="treatment", hue="Gender", data=techmhData, kind="bar", ci=None, size=5, aspect=2, leg
g.set_xticklabels(o)
plt.title('Probability of a Mental Health Disorder')
plt.ylabel('Probability x 100')
plt.xlabel('Care options')

#replace legend labels
new_labels = labelDict['label_Gender']
for t, l in zip(g._legend.texts, new_labels): t.set_text(l)

#positioning the legend
g.fig.subplots_adjust(top=0.9,right=0.8)
plt.show()
```

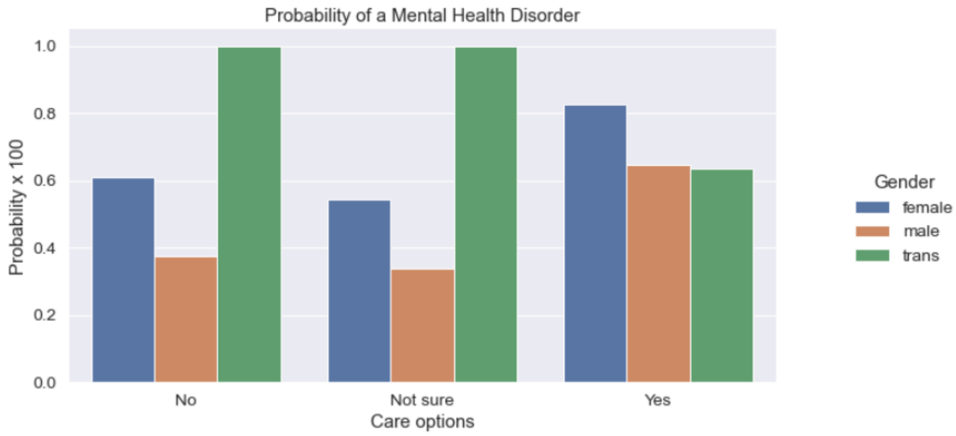


Figure 36. *Probability of whether an employee knows their company's mental health care options.*

"Benefits" variable – Do the employers provide mental health benefits?

Figure 37 shows that nearly 80% of the female survey respondents' employers provide mental health benefits. However, a large number of transgender respondent employers do not and a similar number were unsure whether their employer did. Overall, it appears that nearly 60% of employees who are aware of the benefits want to receive treatment. These findings are concerning and shows how critical it is that companies continue to work on providing good benefits to employees so that they can maintain and improve their mental health disorders, as this will affect their productivity and absenteeism at work.

```
#barplot to show probabilities for "benefits"
o = labelDict['label_benefits']
g = sns.factorplot(x="care_options", y="treatment", hue="Gender", data=techmhData, kind="bar", ci=None, size=5, aspect=2, leg
g.set_xticklabels(o)
plt.title('Probability of a Mental Health Disorder')
plt.ylabel('Probability x 100')
plt.xlabel('Benefits')

#replace legend labels
new_labels = labelDict['label_Gender']
for t, l in zip(g._legend.texts, new_labels): t.set_text(l)

#positioning the legend
g.fig.subplots_adjust(top=0.9,right=0.8)
plt.show()
```

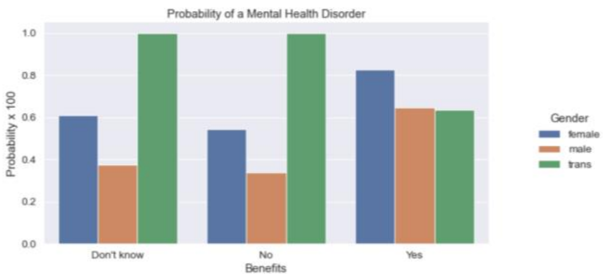


Figure 37. *Probability of whether an employer provides mental health benefits.*

The findings suggest that companies must be aware that gender and family history have a significant impact on whether or not employees seek treatment. As a result, if a company seeks to provide their employees with more support, they must first examine the employee's mental state, because different identities and backgrounds can determine different needs. Age can also be a trigger, and because most of them are young, there is a good chance they are be willing to seek therapy if they are made aware of it. However, employees who identify as transgender may require additional support and encouragement to do so from their employers.

Data pre-processing – Scaling and Fitting

Since the "Age" variable has such a wide range of values, scaling is used, as shown in Figure 38, to make the data points more generalised and lower/normalize the distance between them. Furthermore, because neural networks will be applied later, having an independent variable with a wide range of values could result in a large loss during training and testing, which would cause the learning process to be unstable.

```
#scaling "Age"
scaler = MinMaxScaler()
techmhData['Age'] = scaler.fit_transform(techmhData[['Age']])
techmhData.head()
```

	Age	Gender	Country	self_employed	family_history	treatment	work_interfere	no_employees	remote_work	tech_company	...	leave	mental_heal
0	0.431818	0	44	0	0	1	2	4	0	1	...	2	
1	0.590909	1	44	0	0	0	3	5	0	0	...	0	
2	0.318182	1	6	0	0	0	3	4	0	1	...	1	
3	0.295455	1	43	0	1	1	2	2	0	1	...	1	
4	0.295455	1	44	0	0	0	1	1	1	1	...	0	

5 rows × 25 columns

Figure 38. *Scaling the "Age" variable.*

We then split the dataset into two separate sets: training and test set, as displayed in Figure 39, to enhance the performance of the machine learning model.

```
#splitting the dataset

#define X and y
feature_cols = ['Age', 'Gender', 'family_history', 'benefits', 'care_options', 'anonymity', 'leave', 'work_interfere']
X = techmhData[feature_cols]
y = techmhData.treatment

#split X and y into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)

#create dictionaries for final graph
#use: methodDict['Stacking'] = accuracy_score
methodDict = {}
rmseDict = ()
```

Figure 39. *Splitting the dataset*

Feature Selection (feature importance)

When it comes to prediction, feature selection is critical, as it is the foundation for dimensionality reduction, which allows our model to predict accurately and efficiently. Feature importance is a built-in class for tree-based models, so we will calculate the most relevant and valuable features we need to develop the prediction model using the extra tree classifier, as shown in Figure 40.

```
#build a forest and compute the feature importances
forest = ExtraTreesClassifier(n_estimators=250,
                              random_state=0)

forest.fit(X, y)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

labels = []
for f in range(X.shape[1]):
    labels.append(feature_cols[f])
```

Figure 40. *Building a forest for feature selection.*

A 'forest' of ensemble decision trees is built, as presented in Figure 41, plotting our most important features.

```
#plot the feature importances of the forest
plt.figure(figsize=(12,8))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices],
       color="r", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), labels, rotation='vertical')
plt.xlim([-1, X.shape[1]])
plt.show()
```

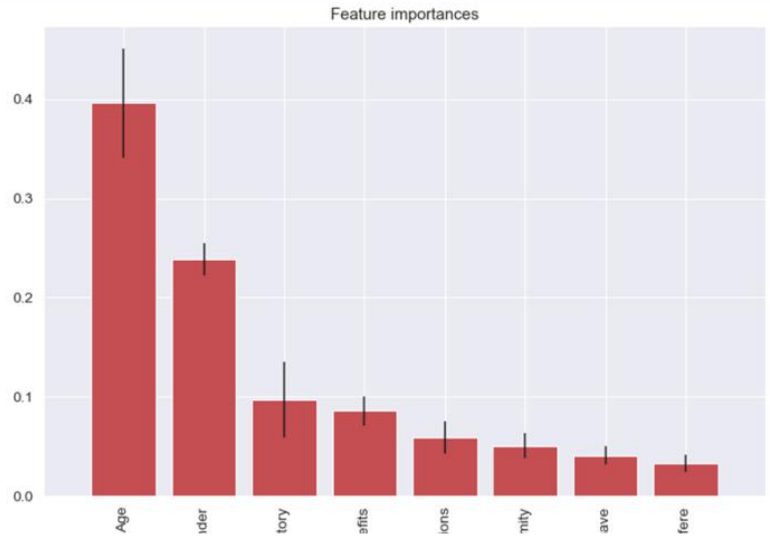


Figure 41. *Plotting feature importances.*

3. Model training, evaluation and testing

Model training

Evaluating a Classification Model

The classification model, as shown in Figure 42, is then built with the aim of drawing a conclusion from the training input values. The main goal is to figure out which class or category the new data belongs in.

```
#this function will evaluate:
#classification accuracy, null accuracy, percentage of ones, percentage of zeros, confusion matrix, false positive rate, prec
def evalClassModel(model, y_test, y_pred_class, plot=False):
#Classification accuracy: percentage of correct predictions
# calculate accuracy
    print('Accuracy:', metrics.accuracy_score(y_test, y_pred_class))

#null accuracy: accuracy that could be achieved by always predicting the most frequent class
#examine the class distribution of the testing set (using a Pandas Series method)
    print('Null accuracy:\n', y_test.value_counts())

#calculate the percentage of ones
    print('Percentage of ones:', y_test.mean())

#calculate the percentage of zeros
    print('Percentage of zeros:',1 - y_test.mean())

#comparing the true and predicted response values
    print('True:', y_test.values[0:25])
    print('Pred:', y_pred_class[0:25])

##^ classification accuracy ref: Ng, R., (2022). Machine Learning with Scikit-Learn - Evaluating a Classification Model. [onl
```

Figure 42. *Building the classification model.*

We evaluate the performance of this classification model using metrics and methods including: confusion matrix, classification accuracy (in Figure 44, which counts how many observations were correctly classified), precision (in Figure 44) and ROC curve (in Figure 45).

The confusion matrix metric, shown in Figure 43, compares the observed and predicted outcome values and displays the number of correct and incorrect predictions by kind of outcome.

There are four categories associated with the observed and predicted outcome values: TP – True Positive, TN – True Negative, FP – False Positive, FN – False Negative.

```
#Confusion matrix

#save confusion matrix and slice into four pieces
    confusion = metrics.confusion_matrix(y_test, y_pred_class)
#[row, column]
    TP = confusion[1, 1]
    TN = confusion[0, 0]
    FP = confusion[0, 1]
    FN = confusion[1, 0]

#visualize Confusion Matrix
    sns.heatmap(confusion,annot=True,fmt="d")
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```

Figure 43. *Producing the confusion matrix.*

As shown in Figure 44, the following metrics are applied:

- **Precision** displays the accuracy of a predicted positive outcome by quantifying the number of correct positive predictions made.
  Precision = TruePositives/(TruePositives + FalsePositives)

- The model's ability to predict the true positive rate is measured by **sensitivity**.
  Sensitivity = TruePositives/(TruePositives + FalseNegatives)

- The model's ability to predict the True Negative Rate is measured by its **specificity**.
  Specificity = TrueNegatives/(TrueNegatives + FalseNegatives)

```
#metrics computed from a confusion matrix

#Classification Accuracy: Overall, how often is the classifier correct?
    accuracy = metrics.accuracy_score(y_test, y_pred_class)
    print('Classification Accuracy:', accuracy)

#Classification Error: Overall, how often is the classifier incorrect?
    print('Classification Error:', 1 - metrics.accuracy_score(y_test, y_pred_class))

#False Positive Rate: When the actual value is negative, how often is the prediction incorrect?
    false_positive_rate = FP / float(TN + FP)
    print('False Positive Rate:', false_positive_rate)

#Precision: When a positive value is predicted, how often is the prediction correct?
    print('Precision:', metrics.precision_score(y_test, y_pred_class))

#first argument is true values, second argument is predicted probabilities
    print('AUC Score:', metrics.roc_auc_score(y_test, y_pred_class))

#calculate cross-validated AUC
    print('Cross-validated AUC:', cross_val_score(model, X, y, cv=10, scoring='roc_auc').mean())

##^ confusion matrix ref: Narkhede, S., 2018. Understanding Confusion Matrix. [online] towardsdatascience.com. & Ng, R., (202
```

```
###
#Adjusting the classification threshold

#print the first 10 predicted responses
#1D array (vector) of binary values (0, 1)
    print('First 10 predicted responses:\n', model.predict(X_test)[0:10])

#print the first 10 predicted probabilities of class membership
    print('First 10 predicted probabilities of class members:\n', model.predict_proba(X_test)[0:10])

#print the first 10 predicted probabilities for class 1
    model.predict_proba(X_test)[0:10, 1]

#store the predicted probabilities for class 1
    y_pred_prob = model.predict_proba(X_test)[:, 1]

    if plot == True:
```

```
#histogram of predicted probabilities
#adjust the font size
        plt.rcParams['font.size'] = 12
#8 bins
        plt.hist(y_pred_prob, bins=8)

#x-axis limit from 0 to 1
        plt.xlim(0,1)
        plt.title('Histogram of predicted probabilities')
        plt.xlabel('Predicted probability of treatment')
        plt.ylabel('Frequency')

#predict treatment if the predicted probability is greater than 0.3
#it will return 1 for all values above 0.3 and 0 otherwise
#results are 2D so we slice out the first column
    y_pred_prob = y_pred_prob.reshape(-1,1)
    y_pred_class = binarize((y_pred_prob, 0.3)[0])


#print the first 10 predicted probabilities
    print('First 10 predicted probabilities:\n', y_pred_prob[0:10])

##^ adjusting the classification threshold ref: Ng, R., (2022). Machine Learning with Scikit-Learn - Evaluating a Classificat
```

Figure 44. *Applying further metrics to examine the performance of the classification model.*

The ROC curve, as shown in Figure 45, is used as its graph measurement visualises the exchange between TP and FP rates using multiple decision thresholds for the prediction model, whilst the AUC measures the classifier's overall performance.

```
###
#ROC Curves and Area Under the Curve (AUC)

##below ROC and Auc ref: Ng, R., (2022). Machine Learning with Scikit-Learn - Evaluating a Classification Model. [online] www
##Patwari, R., 2013. ROC Curves. [online] Youtube.com.

#AUC is the percentage of the ROC plot that is underneath the curve
#higher value = better classifier
    roc_auc = metrics.roc_auc_score(y_test, y_pred_prob)

#first argument is true values, second argument is predicted probabilities
#we pass y_test and y_pred_prob, we do not use y_pred_class, because it will give incorrect results without generating an err
#roc_curve returns 3 objects fpr, tpr, thresholds
#fpr: false positive rate
#tpr: true positive rate
    fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred_prob)
    if plot == True:
        plt.figure()

        plt.plot(fpr, tpr, color='darkorange', label='ROC curve (area = %0.2f)' % roc_auc)
        plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.0])
        plt.rcParams['font.size'] = 12
        plt.title('ROC curve for treatment classifier')
        plt.xlabel('False Positive Rate (1 - Specificity)')
        plt.ylabel('True Positive Rate (Sensitivity)')
        plt.legend(loc="lower right")
        plt.show()

#define a function that accepts a threshold and prints sensitivity and specificity
    def evaluate_threshold(threshold):
#sensitivity: When the actual value is positive, how often is the prediction correct?
#specificity: When the actual value is negative, how often is the prediction correct?print('Sensitivity for ' + str(threshold
        print('Specificity for ' + str(threshold) + ' :', 1 - fpr[thresholds > threshold][-1])
```

```
#define a function that accepts a threshold and prints sensitivity and specificity
    def evaluate_threshold(threshold):
#sensitivity: When the actual value is positive, how often is the prediction correct?
#specificity: When the actual value is negative, how often is the prediction correct?print('Sensitivity for ' + str(threshold
        print('Specificity for ' + str(threshold) + ' :', 1 - fpr[thresholds > threshold][-1])

#one way of setting threshold
    predict_mine = np.where(y_pred_prob > 0.50, 1, 0)
    confusion = metrics.confusion_matrix(y_test, predict_mine)
    print(confusion)

    return accuracy
```

Figure 45. *ROC Curves and AUC and further metrics implementation.*

<u>Hyperparameter Tuning</u>

Hyperparameter tuning is crucial for getting a good performance with the models.

Tuning with cross validation score

We begin by tuning parameters using cross validation, as shown in Figure 46, since our dataset is not too large. Cross validation is a technique for estimating the performance of machine learning models when making predictions on data that was not used during training because it provides a more accurate prediction of out-of-sample performance than train/test split and reduces the variance of a single train/test split trial.

```python
def tuningCV(knn):

#search for an optimal value of K for KNN
    k_range = list(range(1, 31))
    k_scores = []
    for k in k_range:
        knn = KNeighborsClassifier(n_neighbors=k)
        scores = cross_val_score(knn, X, y, cv=10, scoring='accuracy')
        k_scores.append(scores.mean())
    print(k_scores)
#plot the value of K for KNN (x-axis) vs the cross-validated accuracy (y-axis)
    plt.plot(k_range, k_scores)
    plt.xlabel('Value of K for KNN')
    plt.ylabel('Cross-Validated Accuracy')
    plt.show()
```

Figure 46. *Tuning with Cross Validation.*

Tuning with Randomized Search CV

Then, using RandomizedSearchCV, a ‘fit’ and ‘score’ method is implemented. To find the best parameters for the model, this method searches a subset of the parameters and selects random combinations of hyperparameters. Through the use of 'n iter,' as shown in Figure 47, RandomizedSearchCV allows us to specify the number of parameter values we want to test.

```python
def tuningRandomizedSearchCV(model, param_dist):
#Searching multiple parameters simultaneously
#n_iter controls the number of searches
    rand = RandomizedSearchCV(model, param_dist, cv=10, scoring='accuracy', n_iter=10, random_state=5)
    rand.fit(X, y)
    rand.cv_results_

#examine the best model
    print('Rand. Best Score: ', rand.best_score_)
    print('Rand. Best Params: ', rand.best_params_)

#run RandomizedSearchCV 20 times (with n_iter=10) and record the best score
    best_scores = []
    for _ in range(20):
        rand = RandomizedSearchCV(model, param_dist, cv=10, scoring='accuracy', n_iter=10)
        rand.fit(X, y)
        best_scores.append(round(rand.best_score_, 3))
    print(best_scores)

##^ ref: Markham, K., 2021. Efficiently searching for optimal tuning parameters (video #8). [online] GitHub. & Data School, 2
```

Figure 47. *Tuning with RandomizedSearchCV.*

Tuning with searching multiple parameters simultaneously

```python
def tuningMultParam(knn):

#searching multiple parameters simultaneously
#define the parameter values that should be searched
    k_range = list(range(1, 31))
    weight_options = ['uniform', 'distance']

#create a parameter grid: map the parameter names to the values that should be searched
    param_grid = dict(n_neighbors=k_range, weights=weight_options)
    print(param_grid)

#instantiate and fit the grid
    grid = GridSearchCV(knn, param_grid, cv=10, scoring='accuracy')
    grid.fit(X, y)

#view the complete results
    print(grid.grid_scores_)

#examine the best model
    print('Multiparam. Best Score: ', grid.best_score_)
    print('Multiparam. Best Params: ', grid.best_params_)

##^ ref: Markham, K., 2021. Efficiently searching for optimal tuning parameters (video #8). [online] GitHub. & Data School, 2
```

Figure 48. *Tuning with searching multiple parameters.*

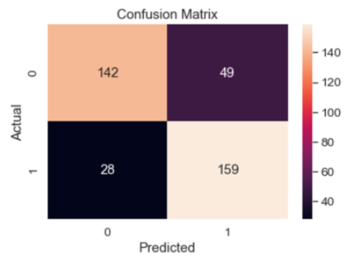
Logistic Regression model training and results

Logistic Regression is a supervised machine learning classification algorithm. It is best used for predictive analysis on a target variable. As shown in Figure 49, we train a logistic regression model on the training set.

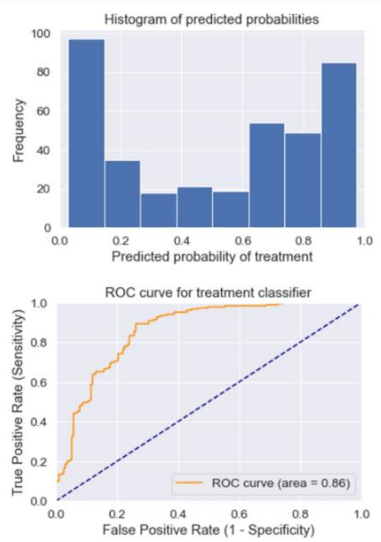
```
def logisticRegression():
#train a logistic regression model on the training set
    logreg = LogisticRegression()
    logreg.fit(X_train, y_train)

#make class predictions for the testing set
    y_pred_class = logreg.predict(X_test)

    print('### Logistic Regression ###')

    accuracy_score = evalClassModel(logreg, y_test, y_pred_class, True)

#data for final graph
    methodDict['Log. Regres.'] = accuracy_score * 100
```

```
logisticRegression()
```

```
### Logistic Regression ###
Accuracy: 0.7962962962962963
Null accuracy:
 0    191
1    187
Name: treatment, dtype: int64
Percentage of ones: 0.4947089947089947
Percentage of zeros: 0.5052910052910053
True: [0 0 0 0 0 0 0 0 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 0]
Pred: [1 0 0 0 1 1 0 1 0 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0 0]
```




```
Classification Accuracy: 0.7962962962962963
Classification Error: 0.20370370370370372
False Positive Rate: 0.25654450261780104
Precision: 0.7644230769230769
AUC Score: 0.7968614385306716
Cross-validated AUC: 0.8753623882722146
First 10 predicted responses:
 [1 0 0 0 1 1 0 1 0 1]
First 10 predicted probabilities of class members:
 [[0.09193053 0.90806947]
```




```
[[142  49]
 [ 28 159]]
```

Figure 49. *Training with Logistic Regression model.*

The output is shown in Figure 49, were:

The Accuracy for this model is – 79%
Precision – 76%
AUC Score – 79%

KnearestNeighbors model training and results

K-nearest neighbours is a supervised machine learning algorithm that can solve classification and regression issues as well as predict the values of new data points. It is the next model we train on the training set, as shown in Figure 50.

```
def Knn():
#calculating the best parameters
    knn = KNeighborsClassifier(n_neighbors=5)

#define the parameter values that should be searched
    k_range = list(range(1, 31))
    weight_options = ['uniform', 'distance']

#specify "parameter distributions" rather than a "parameter grid"
    param_dist = dict(n_neighbors=k_range, weights=weight_options)
    RandomizedSearchCV(knn, param_dist)

#train a KNeighborsClassifier model on the training set
    knn = KNeighborsClassifier(n_neighbors=27, weights='uniform')
    knn.fit(X_train, y_train)

#make class predictions for the testing set
    y_pred_class = knn.predict(X_test)

    print('### KNeighborsClassifier ###')

    accuracy_score = evalClassModel(knn, y_test, y_pred_class, True)

#data for final graph
    methodDict['KNN'] = accuracy_score * 100

##^ ref: Markham, K., 2021. Efficiently searching for optimal tuning parameters (video #8). [online] GitHub.
```

```
Knn()
```

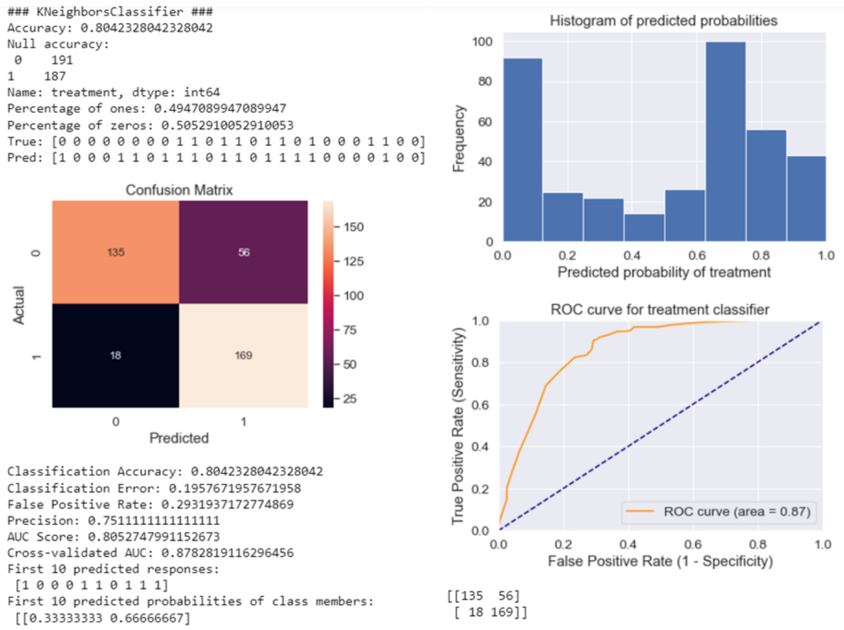


Figure 50. *Training with KNN model.*

The output is shown in Figure 50, were:

The Accuracy for this model is – 80%
Precision – 75%
AUC Score – 80%

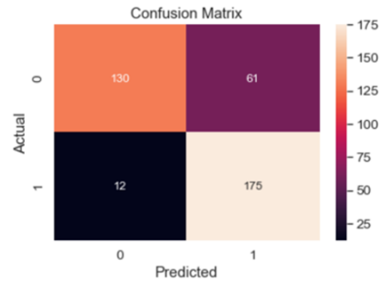
Decision Tree model training and results

The Decision Tree is a supervised machine learning algorithm that relies on a set of rules to make predictions. This algorithm will be used to create our next training model, as shown in Figure 51.

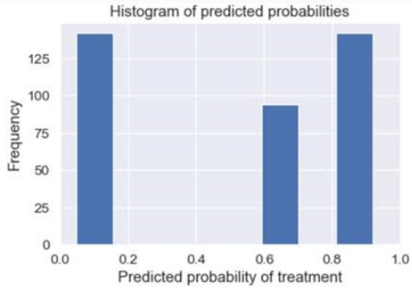
```
def treeClassifier():
#calculating the best parameters
    tree = DecisionTreeClassifier()
    featuresSize = feature_cols.__len__()
    param_dist = {"max_depth": [3, None],
              "max_features": randint(1, featuresSize),
              "min_samples_split": randint(2, 9),
              "min_samples_leaf": randint(1, 9),
              "criterion": ["gini", "entropy"]}
    RandomizedSearchCV(tree, param_dist)

#train a decision tree model on the training set
    tree = DecisionTreeClassifier(max_depth=3, min_samples_split=8, max_features=6, criterion='entropy', min_samples_leaf=7)
    tree.fit(X_train, y_train)

#make class predictions for the testing set
    y_pred_class = tree.predict(X_test)

    print('### Tree classifier ###')

    accuracy_score = evalClassModel(tree, y_test, y_pred_class, True)

#data for final graph
    methodDict['Tree clas.'] = accuracy_score * 100
```

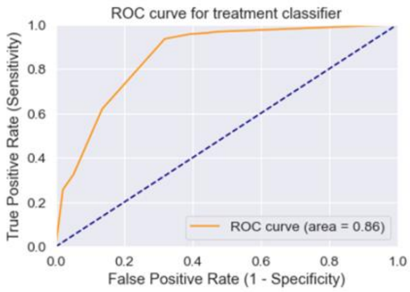
```
treeClassifier()
```

```
### Tree classifier ###
Accuracy: 0.8068783068783069
Null accuracy:
 0    191
1    187
Name: treatment, dtype: int64
Percentage of ones: 0.4947089947089947
Percentage of zeros: 0.5052910052910053
True: [0 0 0 0 0 0 0 0 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 0]
Pred: [1 0 0 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0 0]
```




```
Classification Accuracy: 0.8068783068783069
Classification Error: 0.19312169312169314
False Positive Rate: 0.3193717277486911
Precision: 0.7415254237288136
AUC Score: 0.8082285746283282
Cross-validated AUC: 0.8796535168726989
First 10 predicted responses:
 [1 0 0 0 1 1 0 1 1 1]
First 10 predicted probabilities of class members:
 [[0.18       0.82       ]
```






```
[[130  61]
 [ 12 175]]
```

Figure 51. *Training with Decision Tree model.*

The output is shown in Figure 51, were:

The Accuracy for this model is – 80%
Precision – 74%
AUC Score – 80%

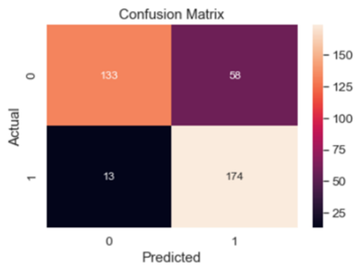
Random Forest model training, results

Random Forest is a popular supervised machine learning algorithm for classification and regression tasks. It can handle large datasets efficiently and its algorithm is known for producing a higher level of accuracy in predicting outcomes. We train this model on the training set, as shown in Figure 52.

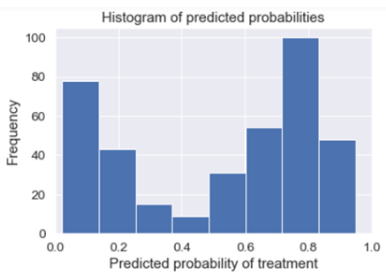
```
def randomForest():
#calculating the best parameters
    forest = RandomForestClassifier(n_estimators = 20)

    featuresSize = feature_cols.__len__()
    param_dist = {"max_depth": [3, None],
              "max_features": randint(1, featuresSize),
              "min_samples_split": randint(2, 9),
              "min_samples_leaf": randint(1, 9),
              "criterion": ["gini", "entropy"]}
    tuningRandomizedSearchCV(forest, param_dist)

#building and fitting my_forest
    forest = RandomForestClassifier(max_depth = None, min_samples_leaf=8, min_samples_split=2, n_estimators = 20, random_stat
    my_forest = forest.fit(X_train, y_train)

#make class predictions for the testing set
    y_pred_class = my_forest.predict(X_test)

    print('### Random Forests ###')

    accuracy_score = evalClassModel(my_forest, y_test, y_pred_class, True)

#data for final graph
    methodDict['R. Forest'] = accuracy_score * 100

    return forest
```

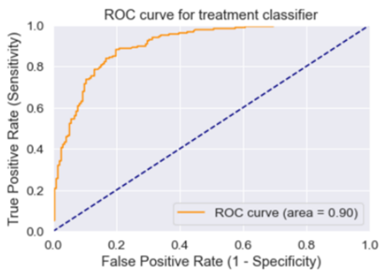
```
model_randomforest=randomForest()
```

```
### Random Forests ###
Accuracy: 0.8121693121693122
Null accuracy:
 0    191
1    187
Name: treatment, dtype: int64
Percentage of ones: 0.4947089947089947
Percentage of zeros: 0.5052910052910053
True: [0 0 0 0 0 0 0 0 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 0]
Pred: [1 0 0 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 0 0 0 1 0 0]
```




```
Classification Accuracy: 0.8121693121693122
Classification Error: 0.1878306878306878
False Positive Rate: 0.3036649214659686
Precision: 0.75
AUC Score: 0.8134081809782457
Cross-validated AUC: 0.8934280651104528
First 10 predicted responses:
 [1 0 0 0 1 1 0 1 1 1]
First 10 predicted probabilities of class members:
 [[0.2555794  0.7444206 ]
```






```
[[133  58]
 [ 13 174]]
```

Figure 52. *Training with Random Forest model.*

The output is shown in Figure 52, were:

The Accuracy for this model is – 81%
Precision – 75%
AUC Score – 81%

Discussion on Models

So far, 4 models have been trained to predict whether an employee may require mental health treatment. The best performance insights of these models has been the Random Forest model. Furthermore, when the AUC score (area under the curve) is considered, this model performs the best and is the highest among the others. However, among the other three models, its confusion matrix Type 1 error is the second highest, which poses a risk because the system could make a false claim about an employee should they end up not requiring mental health treatment. However, this would not be a fatal risk because our system only assists employers in determining what factors influence an employee's decision to seek mental health treatment and then supporting their employees in whatever way is appropriate; it does not decide whether or not an employee **should** seek mental health treatment, in such a way a medical health care system does. Consequently, with a value of 0.81, this model's accuracy outweighs that of the other models, trumping its current suitability for the position.

Predicting with Neural Network

Neural Networks are a set of algorithms that recognise patterns and correlations in a set of data, clustering and classifying them in order to learn and improve. Because neural networks can adapt to changing input, they can produce the best possible outcome without requiring the output parameters to be changed.

As shown in Figure 53, we apply Neural Networks to our system as it is a great for predictive analysis due to its hidden layers, which it uses to make predictions more accurate.

```
# !pip install --upgrade tensorflow-estimator==2.6.0
```

```
#creating input functions
import tensorflow as tf
import argparse


batch_size = 100
train_steps = 1000

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)

def train_input_fn(features, labels, batch_size):
    """An input function for training"""
#convert the inputs to a dataset
    dataset = tf.data.Dataset.from_tensor_slices((dict(features), labels))

# Shuffle, repeat, and batch the examples.
    return dataset.shuffle(1000).repeat().batch(batch_size)

##ref: Morgan, A., 2018. A Developer's intro to TensorFlow and Keras. [online] Scott Logic. & Hui, J., 2017. "TensorFlow Esti

def eval_input_fn(features, labels, batch_size):
    """An input function for evaluation or prediction"""
    features=dict(features)
    if labels is None:
        # No labels, use only features.
        inputs = features
    else:
        inputs = (features, labels)

#convert the inputs to a dataset
    dataset = tf.data.Dataset.from_tensor_slices(inputs)

#batch the examples
    assert batch_size is not None, "batch_size must not be None"
    dataset = dataset.batch(batch_size)

#return the dataset
    return dataset
```

```
#define Tensorflow feature columns
age = tf.feature_column.numeric_column("Age")
gender = tf.feature_column.numeric_column("Gender")
family_history = tf.feature_column.numeric_column("family_history")
benefits = tf.feature_column.numeric_column("benefits")
care_options = tf.feature_column.numeric_column("care_options")
anonymity = tf.feature_column.numeric_column("anonymity")
leave = tf.feature_column.numeric_column("leave")
work_interfere = tf.feature_column.numeric_column("work_interfere")
feature_columns = [age, gender, family_history, benefits, care_options, anonymity, leave, work_interfere]
```

Figure 53. *Training with Neural Networks using Tensorflow.*

In Figure 54, we import DNNClassifier from the Tensorflow library. DNN stands for Deep Neural Network involves building two or more processing layers between the input and output layers, (aiming to mimic the information processing of the human brain) that is trained on a collection of labelled data to accomplish classification.

```
# from tensorflow.compat.v1.estimator.experimental import dnn_logit_fn_builder
# from tensorflow_estimator.python.estimator.canned.dnn import dnn_logit_fn_builder
# !pip install tf-nightly
```

```
#using tf.estimator.DNNClassifier for deep models that perform multi-class classification to predict

#build a DNN with 2 hidden layers and 10 nodes in each hidden layer.
model = tf.estimator.DNNClassifier(feature_columns=feature_columns,
                                    hidden_units=[10, 10],
                                    optimizer=tf.keras.optimizers.Adagrad(
                                      learning_rate=0.1,
#                                       l1_regularization_strength=0.001
                                    ))

##^ tensorflow ref: W3cub, 2020. TensorFlow Guide - W3cubDocs. [online] Docs.w3cub.com. & TensorFlow, 2022. tf.estimator.DNNC
```

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: C:\Users\44792\AppData\Local\Temp\tmpngzao09x
INFO:tensorflow:Using config: {'_model_dir': 'C:\\Users\\44792\\AppData\\Local\\Temp\\tmpngzao09x', '_tf_random_seed': None,
'_save_summary_steps': 100, '_save_checkpoints_steps': None, '_save_checkpoints_secs': 600, '_session_config': allow_soft_pl
acement: true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_step_count_steps': 100, '_train_distribute': Non
e, '_device_fn': None, '_protocol': None, '_eval_distribute': None, '_experimental_distribute': None, '_experimental_max_wor
ker_delay_secs': None, '_session_creation_timeout_secs': 7200, '_checkpoint_save_graph_def': True, '_service': None, '_clust
er_spec': ClusterSpec({}), '_task_type': 'worker', '_task_id': 0, '_global_id_in_cluster': 0, '_master': '', '_evaluation_ma
ster': '', '_is_chief': True, '_num_ps_replicas': 0, '_num_worker_replicas': 1}
```

Figure 54. *Using the DNNClassifier to predict.*

Evaluation

The test set accuracy using TensorFlow was 0.81%, as seen in Figure 55. The closer the model's performance gets to 1, the better it is, therefore scoring in the 80s range is great and realistic.

```
#evaluate the model
eval_result = model.evaluate(
    input_fn=lambda:eval_input_fn(X_test, y_test, batch_size))

print('\nTest set accuracy: {accuracy:0.2f}\n'.format(**eval_result))

#data for final graph
accuracy = eval_result['accuracy'] * 100
methodDict['NN DNNClasif.'] = accuracy
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2022-04-23T15:10:36
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from C:\Users\44792\AppData\Local\Temp\tmpngzao09x\model.ckpt-1000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Inference Time : 0.56791s
INFO:tensorflow:Finished evaluation at 2022-04-23-15:10:36
INFO:tensorflow:Saving dict for global step 1000: accuracy = 0.8121693, accuracy_baseline = 0.505291, auc = 0.88840044, auc_
precision_recall = 0.85664, average_loss = 0.4219399, global_step = 1000, label/mean = 0.49470899, loss = 0.4234065, precisi
on = 0.75, prediction/mean = 0.50508994, recall = 0.93048126
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 1000: C:\Users\44792\AppData\Local\Temp\tmpngzao09x\model.c
kpt-1000

Test set accuracy: 0.81
```

Figure 55. *Evaluating the model.*

Testing

Now, as presented in Figure 56, we start testing the trained model, using three frames: Index (rows/columns), Prediction, and Expected.

```
#using the trained model to predict whether an employee will need treatment or not
predictions = list(model.predict(input_fn=lambda:eval_input_fn(X_train, y_train, batch_size=batch_size)))
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from C:\Users\44792\AppData\Local\Temp\tmpngzao09x\model.ckpt-1000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
```

```
#generate predictions from the model
template = ('\nIndex: "{}", Prediction is "{}" ({:.1f}%), expected "{}"')

#dictionary for predictions
col1 = []
col2 = []
col3 = []


for idx, input, p in zip(X_train.index, y_train, predictions):
    v  = p["class_ids"][0]
    class_id = p['class_ids'][0]
    probability = p['probabilities'][class_id] #probability

#adding to dataframe
    col1.append(idx) #index
    col2.append(v) #prediction
    col3.append(input) #expecter


 #print(template.format(idx, v, 100 * probability, input))
```

Figure 56. *Testing on the trained model to predict.*

```
results = pd.DataFrame({'index':col1, 'prediction':col2, 'expected':col3})
results.head()
```

	index	prediction	expected
0	929	0	0
1	901	1	1
2	579	1	1
3	367	1	1
4	615	1	1

Figure 57. *Outcome of the trained models prediction.*

4. Results and discussion

As shown above in Figure 57, the trained model's prediction in the output array matches the expected results. Reiterating that the model is robust and accurate.

Since the NN DNN Classifier and Random Forest both have the same percentage of success (shown below in Figure 61), they are both considered the best solution for this project and can be used to make predictions. In Figure 58, we use Random Forest to make predictions about whether a tech employee may require mental health treatment.

```
#generate predictions with the best method
clf = model_randomforest
clf.fit(X, y)
dfTestPredictions = clf.predict(X_test)
```

Figure 58. *Using the Random Forest model to predict treatment.*

```
#write predictions to csv file
#there's no significative field so the index is saved
results = pd.DataFrame({'Index': X_test.index, 'Treatment': dfTestPredictions})
#save to file

#this file will be visible after publishing in the output section
results.to_csv('results.csv', index=False)
results
```

	Index	Treatment
0	5	1
1	494	0
2	52	0
3	984	0
4	186	0
...	...	...
373	1084	1
374	506	0
375	1142	0
376	1124	0
377	689	1

378 rows × 2 columns

Figure 59. *Outcome of Random Forest prediction.*

Results Comparison

```
def plotSuccess():
    s = pd.Series(methodDict)
    s = s.sort_values(ascending=False)
    plt.figure(figsize=(12,8))
#colors
    ax = s.plot(kind='bar')
    for p in ax.patches:
        ax.annotate(str(round(p.get_height(),2)), (p.get_x() * 1.005, p.get_height() * 1.005))
    plt.ylim([70.0, 90.0])
    plt.xlabel('Method')
    plt.ylabel('Percentage')
    plt.title('Success of methods')

    plt.show()
```

```
plotSuccess()
```

Figure 60. *Code to plot of the Success of the Methods.*

A results comparison, as shown in Figure 61, across the models built:

1. The most powerful models in this project were **NN DNN Classifier** (Deep Neural Networks) and **Random Forest**, which tied for first place and outperformed the other three models.
2. **Tree Classifier** has performed admirably in the prediction.
3. The **KNN** model lags behind the **Tree Classifier** in prediction, by a small margin.
4. In comparison to the **NN DNN Classifier** and **Random Forest**, **Logistic Regression** performed significantly worse due to underfitting.
5. The order of the best algorithms:

NN DNN Classifier (Deep Neural Networks) **& Random Forest** > **Tree Classifier** > **KNN**.

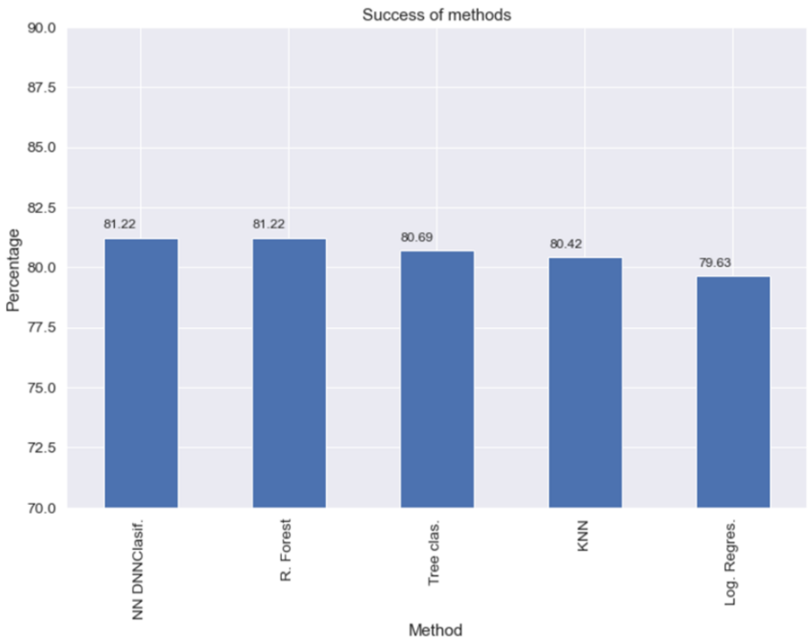


Figure 61. *Plot of the Success of the Methods.*

<u>**Analysis of AI project life cycle**</u>

Iyer (2021) outlines five stages of the AI Project Life Cycle: **Problem Scoping, Data Acquisition, Data Exploration, Modelling** and **Evaluation**. AI ethics is a set of beliefs, concepts, and methodologies that apply widely recognised moral standards to govern moral behaviour in the development and application of AI technologies. Data is an extremely valuable resource, and it is up to the researcher to extract its value and full potential. In this project, the 'Data Acquisition' stage of the cycle reveals areas where AI ethics are necessary. Data Acquisition is the stage in which we gather the information we need and convert it into a visual representation, with this information serving as the system's foundation. This information must be accurate and reliable in order for our system to function properly. The data for our dataset was obtained from a non-profit organisation that shared it publicly in order to gain as many insights from researchers and data scientists as possible, which would contribute to further findings to support the knowledge improvement area of mental health for employees working within technology. As a result, a considerable number of respondents agreed to share their input for the greater benefit, and the investigation's findings should be treated as such, with respect, especially in terms of **data privacy**. It should be noted that data privacy was protected in this way because the employees who contributed were unable to be recognised and were merely labelled as numbers just so that the researchers could identify between them. Any potential **racial bias** was eliminated as a result of the removal of such sensitive and **personal data**, which, if left in place, could have also violated each employee's right to privacy.

Furthermore, an exploratory data analysis was undertaken during the 'Data Exploration' stage, which revealed that each employee's gender played a significant factor in their experiences with mental health while working in tech. There was a discrepancy in the numbers of responses from two gender groups: trans and female employees, indicating that possible **gender bias** may have been present because the number of males who participated in the survey outweighed both of them. As a result, the system *may* contain gender bias, albeit only a <u>*slight*</u> possibility, due to the model learning and duplicating the bias (since it does not know any better). Additionally, the data samples used to train and evaluate algorithmic systems can sometimes be unrepresentative of the populations from which inferences are drawn. If you can imagine, not every employee working in tech around the world has the opportunity to bring their laptops or computers home for leisure. So there is a possibility that the data being fed into the system is slightly tendentious from the outset, which results in a genuine risk of the system producing **discriminatory outcomes**. Although, this was sought to be countered against by collecting data from a global survey with employees from over 25 different countries. More data focusing on trans and female employees in tech, particularly those who work in disadvantaged areas throughout the world, should be collected in the future and then fed back as new data into the model to train with, in order to aid the development of a more gender-equal predictive system.

Conclusion

Using supervised machine learning algorithms, it is possible to predict whether a tech employee will seek mental health treatment. We implemented a machine learning pipeline that included data pre-processing, data visualization, feature selection, model training, model evaluation and model testing, based on the data provided in the dataset.

Random forest and Neural Networks Deep Neural Network Classifier outperformed and produced the best results for the survey dataset, both of which had an accuracy of 81%.

Recommendations

During the EDA, instead of separating the charts based on gender, the data could have been further separated into disorder types, such as anxiety, depression, and so on, to aid in determining which of these disorders affect work productivity, treatment, and use of the wellness programmes available at the workplace. This would also aid in the development of a more accurate prediction as to which of the employees' disorders may be more likely to result in treatment requests.

Furthermore, in one of the charts, there is an outlier (the female 66-100 year old) that led to the conclusion that 100% of women aged 66 to 100 have mental health disorders, so the predictor's outcome could be slightly skewed as a result of this outlier.

Future work

There is increased interest in the LGBTQ+ group, based on current societal issues. Although the number of LGBT responses was small, it can be assumed that more research into the topic would yield some interesting results and new insights. As a marginalised group that is publicly discriminated against around the world, it is reasonable to expect that them to encounter hate speech or disparaging comments in the workplace, which may exacerbate their mental health issues and contribute to their choice to seek treatment or even become diagnosed with more severe mental health disorders.

Additionally, the prediction could be ran again on a newer dataset produced post-COVID, since more employees throughout the world were forced to work remotely full-time owing to the circumstances of COVID-19, which would have had a formidable mental impact on them as their socialisation was reduced and their isolation increased. However this would be very difficult given that the most recent surveys issued include too many missing fields from the respondents feedback as well as a lower number of respondents willing to participate, which would cast doubt on the credibility of the results.

References

Chan, R., 2020. *Meet the developer group trying to break the mental health stigma in tech, where moving fast and breaking things leads developers to burnout and health issues*. [online] Business Insider. Available at: https://www.businessinsider.com/open-sourcing-mental-illness-stigma-tech-2019-12?r=US&IR=T#:~:text=In%202013%2C%20developer%20Ed%20Finkler,issues%20than%20the%20general%20population. [Accessed 1 April 2022].

Data School, 2014. *ROC Curves and Area Under the Curve (AUC) Explained*. [online] Youtube.com. Available at: https://www.youtube.com/watch?v=OAl6eAyP-yo [Accessed 18 April 2022].

Data School, 2019. *Machine learning in Python with scikit-learn (Playlist)*. [online] Youtube.com. Available at: https://www.youtube.com/watch?v=elojMnjn4kk&list=PL5-da3qGB5ICeMbQuqbbCOQWcS6OYBr5A [Accessed 18 April 2022].

Dattani, S., Ritchie, H., and Roser, M., 2018. *Mental Health.* [online] OurWorldInData.org. Available at: https://ourworldindata.org/mental-health [Accessed 1 April 2022]

Dunne, O., and Hewson, H. 2021., *4 Reasons Why the UK Tech Industry is Thriving*. [online] Caminosearch.co.uk. Available at: https://www.caminosearch.co.uk/blog/2021/07/4-reasons-why-the-uk-tech-industry-is-thriving?source=google.com [Accessed 1 April 2022].

FDM, 2016. *7 Reasons Why Tech is the Best Industry for Job Hunters*. [online] fdmgroup.com. Available at: https://www.fdmgroup.com/7-reasons-why-it-is-the-best-industry-for-job-hunters/ [Accessed 1 April 2022].

Global Burden of Disease Collaborative Network (GBD), 2016. *GBD Global Burden of Disease Study 2016*. [online] Seattle, United States: Institute for Health Metrics and Evaluation (IHME), 2017 Available at: http://ghdx.healthdata.org/gbd-results-tool [Accessed 1 April 2022]

Gupta, T., 2018. *Mental Health in the Tech Workplace*. [online] Rstudio-pubs-static.s3.amazonaws.com. Available at: https://rstudio-pubs-static.s3.amazonaws.com/346297_e30216baa3664cd393b87471f3da467a.html [Accessed 2 April 2022].

Hui, J., 2017. *"TensorFlow Estimator"*. [online] Jhui.github.io. Available at: https://jhui.github.io/2017/03/14/TensorFlow-Estimator/ [Accessed 19 April 2022].

Iyer, S., 2021. *Class 9,10 Artificial Intelligence | Unit 2 | AI Project Cycle*. [online] Youtube.com. Available at: https://www.youtube.com/watch?v=V7QzWen9Odk [Accessed 3 May 2022].

Jack, 2020. *Digital tech industry growing six times faster than any other sector – How do we bridge the digital skills gap?*. [online] FE News. Available at: https://www.fenews.co.uk/fe-voices/digital-tech-industry-growing-six-times-faster-than-any-other-sector-how-do-we-bridge-the-digital-skills-gap/ [Accessed 1 April 2022].

Javatpoint., 2019. *Data Preprocessing in Machine Learning*. [online] Javatpoint.com. Available at: https://www.javatpoint.com/data-preprocessing-machine-learning [Accessed 1 April 2022].

Johnson, L., 2019. *Mental Health in Tech Workplace Analysis*. [online] Linkedin.com. Available at: https://www.linkedin.com/pulse/mental-health-tech-workplace-analysis-lance-johnson/ [Accessed 2 April 2022].

Kaggle, 2014. Mental Health in Tech Dataset. [online]. Available at: https://www.kaggle.com/osmi/mental-health-in-tech-survey [Accessed 28 March 2022].

Kalanithi, S., 2021. *Hands on Machine Learning - Chapter 3 - Classification*. [online] Youtube.com. Available at: https://www.youtube.com/watch?v=pAzkdQazqJY [Accessed 17 April 2022].

Larson, K., 2021. *Analyzing Mental Health in the Tech Workplace Survey 2014*. [online] Medium.com. Available at: https://medium.com/@kaylalarson333/ds-unit-1-portfolio-project-ad71fe62cd6c [Accessed 2 April 2022].

Markham, K., 2021. *Efficiently searching for optimal tuning parameters (video #8)*. [online] GitHub. Available at: https://github.com/justmarkham/scikit-learn-videos/blob/master/08_grid_search.ipynb [Accessed 19 April 2022].

Mesquita, D., 2021. *Python AI: How to Build a Neural Network & Make Predictions*. [online] Realpython.com. Available at: https://realpython.com/python-ai-neural-network/ [Accessed 20 April 2022].

Morgan, A., 2018. *A Developer's intro to TensorFlow and Keras*. [online] Scott Logic. Available at: https://blog.scottlogic.com/2018/10/25/a-developers-intro-tensorflow-and-keras.html [Accessed 19 April 2022].

Narkhede, S., 2018. *Understanding Confusion Matrix*. [online] towardsdatascience.com. Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 [Accessed 17 April 2022].

Ng, R., 2022. *Machine Learning with Scikit-Learn - Evaluating a Classification Model*. [online] www.ritchieng.com. Available at: https://www.ritchieng.com/machine-learning-evaluate-classification-model/ [Accessed 17 April 2022].

OSMH, 2014. *OSMH/OSMI Mental Health in Tech Survey*. [online] Osmhhelp.org. Available at: https://osmhhelp.org/research [Accessed 1 April 2022].

Patwari, R., 2013. *ROC Curves*. [online] Youtube.com. Available at: https://www.youtube.com/watch?v=21Igj5Pr6u4 [Accessed 18 April 2022].

TensorFlow, 2022. *tf.estimator.DNNClassifier*. [online] TensorFlow. Available at: https://www.tensorflow.org/api_docs/python/tf/estimator/DNNClassifier [Accessed 19 April 2022].

World Health Organization WHO, 2018. *Mental health: strengthening our response*. [online] Who.int. Available at: https://www.who.int/news-room/fact-sheets/detail/mental-health-strengthening-our-response [Accessed 1 April 2022].

W3cub, 2020. *TensorFlow Guide - W3cubDocs*. [online] Docs.w3cub.com. Available at: https://docs.w3cub.com/tensorflow~guide/get_started/premade_estimators [Accessed 19 April 2022].

Appendix

Appendices 1 – Table with the full length questions asked in the survey.

Timestamp	Date and Time Survey was filled out
Age	
Gender	
Country	
state	If you live in the United States, which state or territory do you live in?
self_employed	Are you self-employed?
family_history	Do you have a family history of mental illness?
treatment	Have you sought treatment for a mental health condition?
work_interfere	If you have a mental health condition, do you feel that it interferes with your work?
no_employees	How many employees does your company or organization have?
remote_work	Do you work remotely (outside of an office) at least 50% of the time?
tech_company	Is your employer primarily a tech company/organization?
benefits	Does your employer provide mental health benefits?

care_options	Do you know the options for mental health care your employer provides?
wellness_program	Has your employer ever discussed mental health as part of an employee wellness program?
seek_help	Does your employer provide resources to learn more about mental health issues and how to seek help?
anonymity	Is your anonymity protected if you choose to take advantage of mental health or substance abuse treatment resources?
leave	How easy is it for you to take medical leave for a mental health condition?
mentalhealthconsequence	Do you think that discussing a mental health issue with your employer would have negative consequences?
physhealthconsequence	Do you think that discussing a physical health issue with your employer would have negative consequences?
coworkers	Would you be willing to discuss a mental health issue with your coworkers?
supervisor	Would you be willing to discuss a mental health issue with your direct supervisor(s)?
mentalhealthinterview	Would you bring up a mental health issue with a potential employer in an interview?
physhealthinterview	Would you bring up a physical health issue with a potential employer in an interview?
mentalvsphysical	Do you feel that your employer takes mental health as seriously as physical health?
obs_consequence	Have you heard of or observed negative consequences for coworkers with mental health conditions in your workplace?
comments	Any additional notes or comments