Configuration Manual

MSc Research Project
Programme Name

Harshita Singh
Student ID: x19196725

School of Computing
National College of Ireland

Supervisor: Dr. Rashmi Gupta

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Harshita Singh
Student ID:	x19196725
Programme:	Data Analytics
Year:	2020-2021
Module:	Research Project
Supervisor:	Dr. Rashmi Gupta
Submission Due Date:	16-Aug-2021
Project Title:	Identification of mental disorder based on the symptoms of the person using machine learning algorithms
Word Count:	XXX
Page Count:	14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	16th August 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Forename Surname
XXX

1 Introduction

The aim of this paper is to provide a quick overview of the measures involved in the implementation of this project. The objective of the research was to assess the effectiveness of a mental disease classifier utilizing a data pre-processing technique with a function selection strategy and a new machine learning approach. The second goal was to compare the performance and assess the performance of the model with few other classifiers. In the remaining section of the manual are referred to the tools and strategies utilized to achieve the defined goals.

2 System Specifications

The system configuration which includes memory and operating system on which this research project has been carried out is mentioned below:

- Operating System: Windows 10 Home
- Installed Memory (RAM): 8.0 GB
- Hard Drive: 1024 GB HDD
- Processor: Intel® Core™ i5-1035G1 CPU @ 1.19GHz

3 Tools and Technologies

Python's programming language was utilized for this project, while Jupyter Notebook was employed as an integrated development environment (IDE). The visualization was done using python alone. The following are the specific versions of the relevant platform/language:

- Python 3.7.2
- Jupyter Notebook Server v. 6.0.2

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		02a75015f7cd845e27b85192bb0ca4cb	22897802	SIG
XZ compressed source tarball	Source release		df6ec36011808205beda239c72f947cb	17042320	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	d8ff07973bc9c009de80c269fd7efcca	34405674	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	0fc95e9f6d6b4881f3b499da338a9a80	27766090	SIG
Windows help file	Windows		941b7d6279c0d4060a927a65dcab88c4	8092167	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	f81568590bef56e5997e63b434664d58	7025085	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	ff258093f0b3953c886192dec9f52763	26140976	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	8de2335249d84fe1eeb61ec25858bd82	1362888	SIG
Windows x86 embeddable zip file	Windows		26881045297dc1883a1d61baffeecaf0	6533256	SIG
Windows x86 executable installer	Windows		38156b62c0cbcb03bfddeb86e66c3a0f	25365744	SIG
Windows x86 web-based installer	Windows		1e6c626514b72e21008f8cd53f945f10	1324648	SIG

Figure 1: environment setup

4 Environment setup

Installation of the appropriate platform and languages is the first and most important step in the implementation of the project. The following URL has been used to download and install Python.

- On the following URL, Jupyter is installed by using the installation guide. We have to execute the command at CMD prompt to launch the Jupyter Notebook.

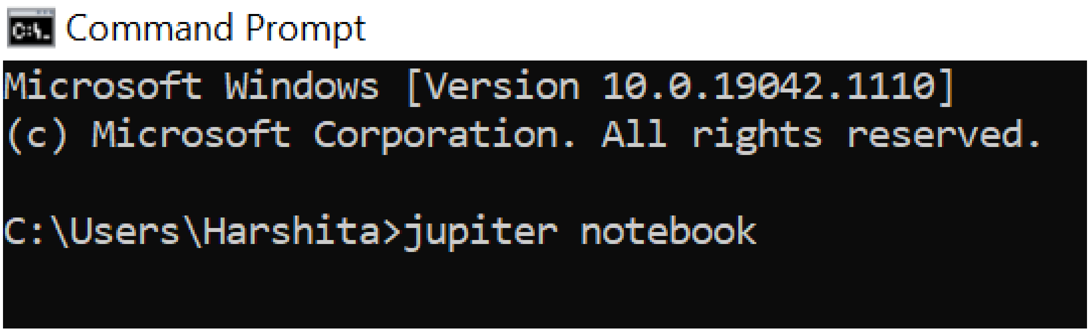


Figure 2: Command prompt

- After completing the project, the results of the different diagrams were displayed, comparisons and selection of features were also displayed in the python itself.

5 Data Collection

Data was obtained for this project via the common repository and public repository kaggle. The description of the data is as follows:

There are 25 columns in the datasets, with 24 columns boolean, with 1 column string. The attributes are:

- feeling nervous

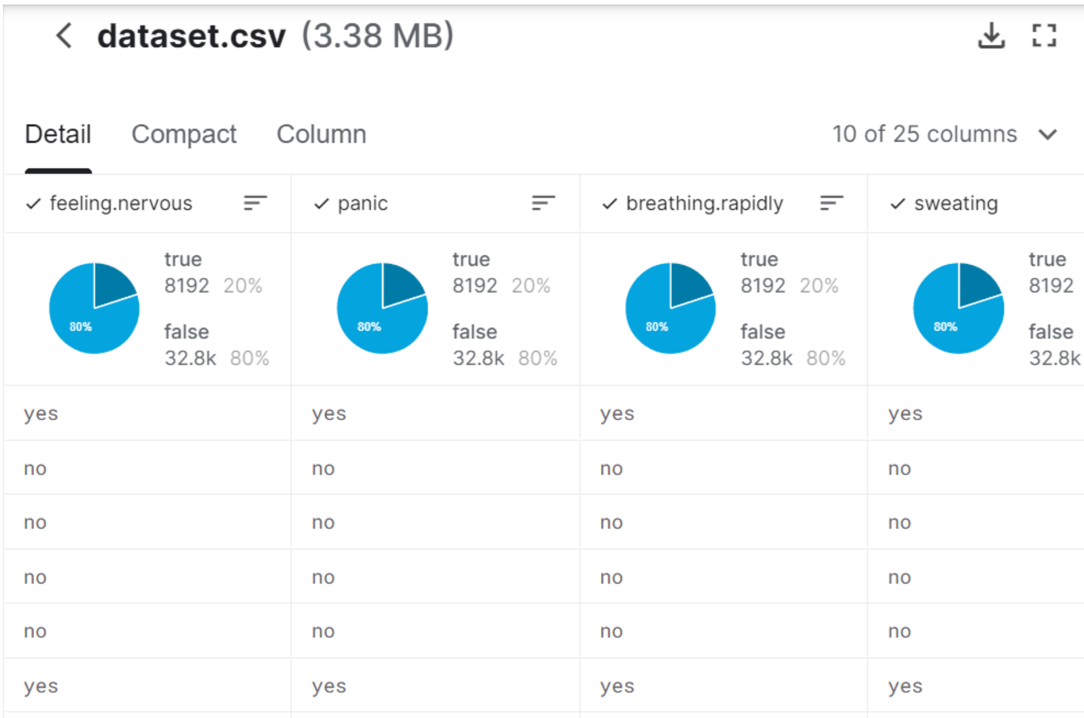


Figure 3: Dataset Description

- panic
- breathing rapidly
- sweating,
- trouble in concentration
- having trouble in sleeping
- having trouble with work
- hopelessness
- anger
- over react
- change in eating
- suicidal thought
- feeling tired
- close friend
- social media addiction
- weight gain

- material possessions
- introvert
- popping up stressful memory
- having nightmares
- avoids people or activities
- feeling negative
- trouble concentrating
- blaming yourself and
- Disorder.

These symptoms are attributes of the dataset based on the disorder of the types of mental illness.

6 Implementation:

The entire code is accessible on the following GitHub repository for this project. The many procedures involved in this project are explained step by step below.

6.1 Data Preparation and Storage

In data preparation has been done collecting the various symptoms and disorders of the person from different websites.

- File was in CSV format available openly in kaggle website which is a open source website.
- For the study mental disorder symptoms based on the type of mental illness are used for the analysis.
- Here we first set the seed for our code since it preserves the data samples and the findings.
- The data file was then saved to Github in CSV format, then imported to GitHub using the following code.

```
In [5]: import numpy as np # linear algebra
        import pandas as pd # data processing

        import matplotlib.pyplot as plt
        import seaborn as sns
```

The command is use to load the csv file by using the function read csv is shown in below image.

```
In [6]: data = pd.read_csv("archive/dataset.csv")
```

Out[2]:

	feeling.nervous	panic	breathing.rapidly	sweating	trouble.in.concentration	having.trouble.in.sleeping	having.trouble.with.work	hopel
0	yes	yes	yes	yes	yes	yes	no	
1	no	no	no	no	no	no	yes	
2	no	no	no	no	no	no	no	
3	no	no	no	no	no	no	no	
4	no	no	no	no	no	no	no	

5 rows × 25 columns

The below figure shows the format of data as data was in string format which was then converted on Boolean format.

The below image shows the shape of data as the data consist of 40960 columns and 25 rows with various symptoms in of mental disorder in it such as panic attacks, breathing trouble, sleeping disorder, stress etc.

```
In [8]: data.shape
Out[8]: (40960, 25)

In [9]: data.columns
Out[9]: Index(['feeling.nervous', 'panic', 'breathing.rapidly', 'sweating',
         'trouble.in.concentration', 'having.trouble.in.sleeping',
         'having.trouble.with.work', 'hopelessness', 'anger', 'over.react',
         'change.in.eating', 'suicidal.thought', 'feeling.tired', 'close.friend',
         'social.media.addiction', 'weight.gain', 'material.possessions',
         'introvert', 'popping.up.stressful.memory', 'having.nightmares',
         'avoids.people.or.activities', 'feeling.negative',
         'trouble.concentrating', 'blamming.yourself', 'Disorder'],
        dtype='object')
```

The below image shows the information of the dataset does not consists of any null values as the dataset was in clean.

The below image shows the information of the dataset does not consists of any null values as the dataset was in clean.It is vital to comprehend the data set before starting our pre-processing stage, so that we are aware of the subsequent actions for cleaning and pre-processing purposes.

The dataset is converted in boolean format as shown in below image which was in string format.

6.2 Exploratory Data Analysis

It is vital that we comprehend the information so that we know how to proceed as part of the cleaning and pre-processing before we begin with our pre-processing phase.

- As the dataset comprised of many variables, few of these variables could possibly be linked. This can lower the performance of the model and increase the time and resources for calculation. The multi-linearity test was thus carried out with the following code.

```
In [10]: data.info()
```

```
Data columns (total 25 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   feeling.nervous                40960 non-null  object
 1   panic                          40960 non-null  object
 2   breathing.rapidly              40960 non-null  object
 3   sweating                       40960 non-null  object
 4   trouble.in.concentration       40960 non-null  object
 5   having.trouble.in.sleeping     40960 non-null  object
 6   having.trouble.with.work       40960 non-null  object
 7   hopelessness                   40960 non-null  object
 8   anger                          40960 non-null  object
 9   over.react                     40960 non-null  object
 10  change.in.eating               40960 non-null  object
 11  suicidal.thought               40960 non-null  object
 12  feeling.tired                  40960 non-null  object
 13  close.friend                   40960 non-null  object
 14  social.media.addiction         40960 non-null  object
 15  weight.gain                    40960 non-null  object
 16  material.possessions           40960 non-null  object
 17  introvert                      40960 non-null  object
 18  popping.up.stressful.memory    40960 non-null  object
 19  having.nightmares              40960 non-null  object
 20  avoids.people.or.activities    40960 non-null  object
 21  feeling.negative               40960 non-null  object
 22  trouble.concentrating          40960 non-null  object
 23  blamming.yourself              40960 non-null  object
 24  Disorder                       40960 non-null  object
dtypes: object(25)
```

In addition, the use of pandas profiling was examined in further depth by individual variables for missing values and skewness. The following is the code for this purpose

6.3 Data Cleaning

We must clean up our data and make changes so that the model can provide optimum performance before continue with the model development step. Variable data types were converted, duplicate rows removed, variables with high multicollinearities dropped, columns with a large number of missing values were removed or missed values imputed using mean in other columns, amongst other things. Different stages involved. In the following pictures, the code for each step is presented.

6.4 Feature Selection

We followed the usage of the selection approach Extra Tree Classifier to reduce characteristics. The packages needed to do this are shown below The following code is listed

7 Modelling

Following the installation of all of the models, their performance was assessed using various indicators. These measures were picked after doing a literature research and were visually contrasted using a bar graph. The code for which is given below-

The comparison ocf acciracy between different models is find out by below code

The below image shows the accuracy of the performance of the model of all the algorithms. The logistic Regression, Decision tree and XGBoost showed the similar accuracy and ffnn and svm achieved the lowest.

```
In [14]: data.head()
```

Out[14]:

	feeling.nervous	panic	breathing.rapidly	sweating	trouble.in.concentration	having.trouble.in.sleeping	having.trouble.with.work	hopelessness	anger	over.rea
0	1	1	1	1	1	1	0	0	0	
1	0	0	0	0	0	0	1	1	1	
2	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	

5 rows × 25 columns

Exploratory Data Analysis

```
In [11]: data['Disorder'].value_counts()

Out[11]: Normal        8192
         Depression    8192
         Anxiety       8192
         Loneliness    8192
         Stress        8192
         Name: Disorder, dtype: int64
```

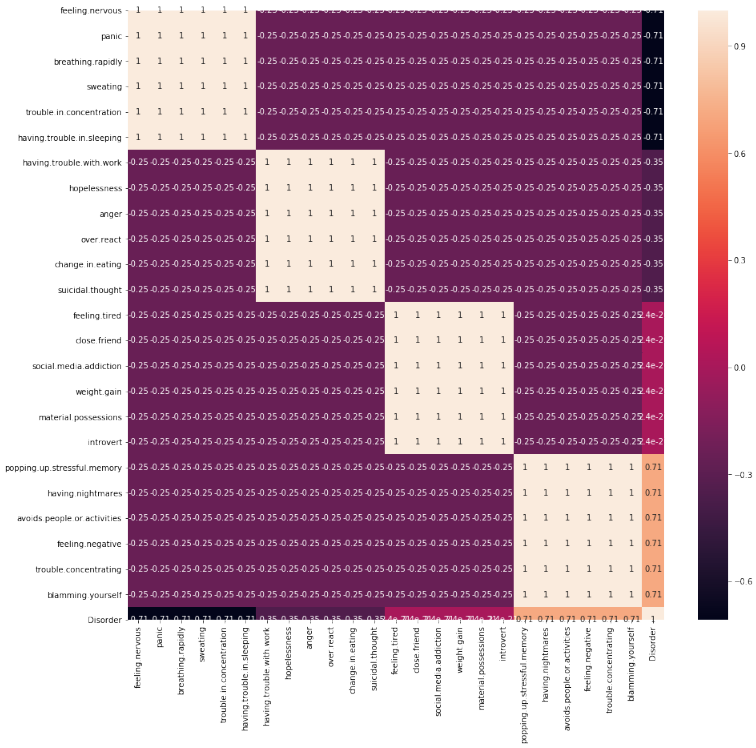
8 Conclusion

The whole implementation procedure of this project has been outlined in a succinct, thorough, and sequential way using the information presented in the preceding parts. The needed packages have also been indicated wherever they were used, and the whole code has been released on GitHub, the URL for which may be found in the Implementation section.

References

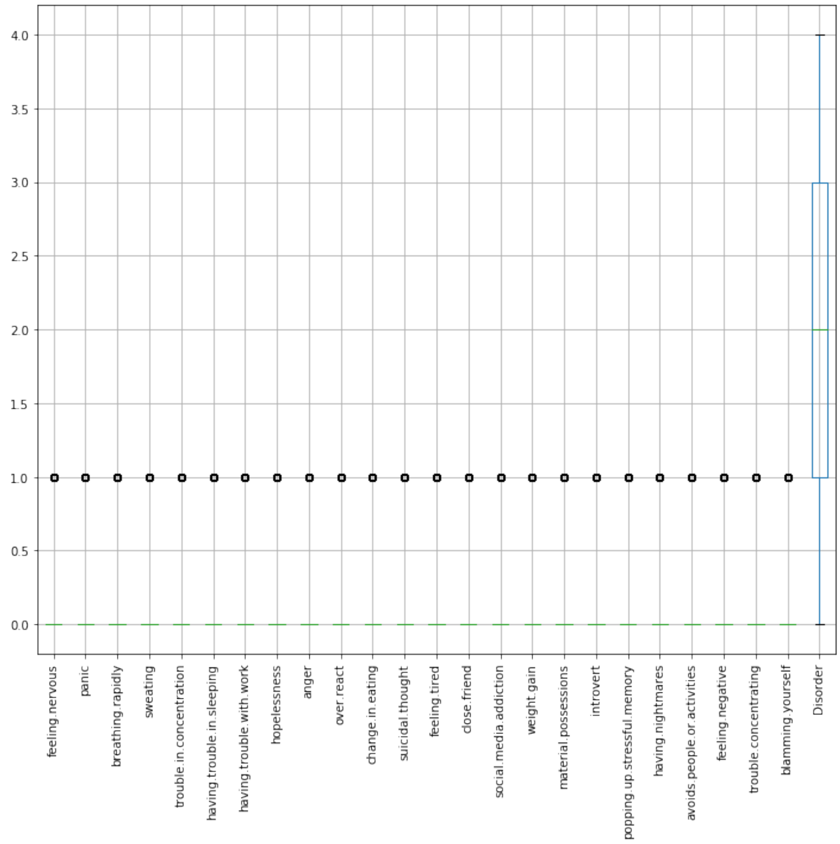
In [15]:
```
plt.figure(figsize = (15,14))
sns.heatmap(data.corr(), annot = True)
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x12993d4f0>



In [16]:
```
data.boxplot(rot = 90, figsize = (12,10))
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x12c617400>

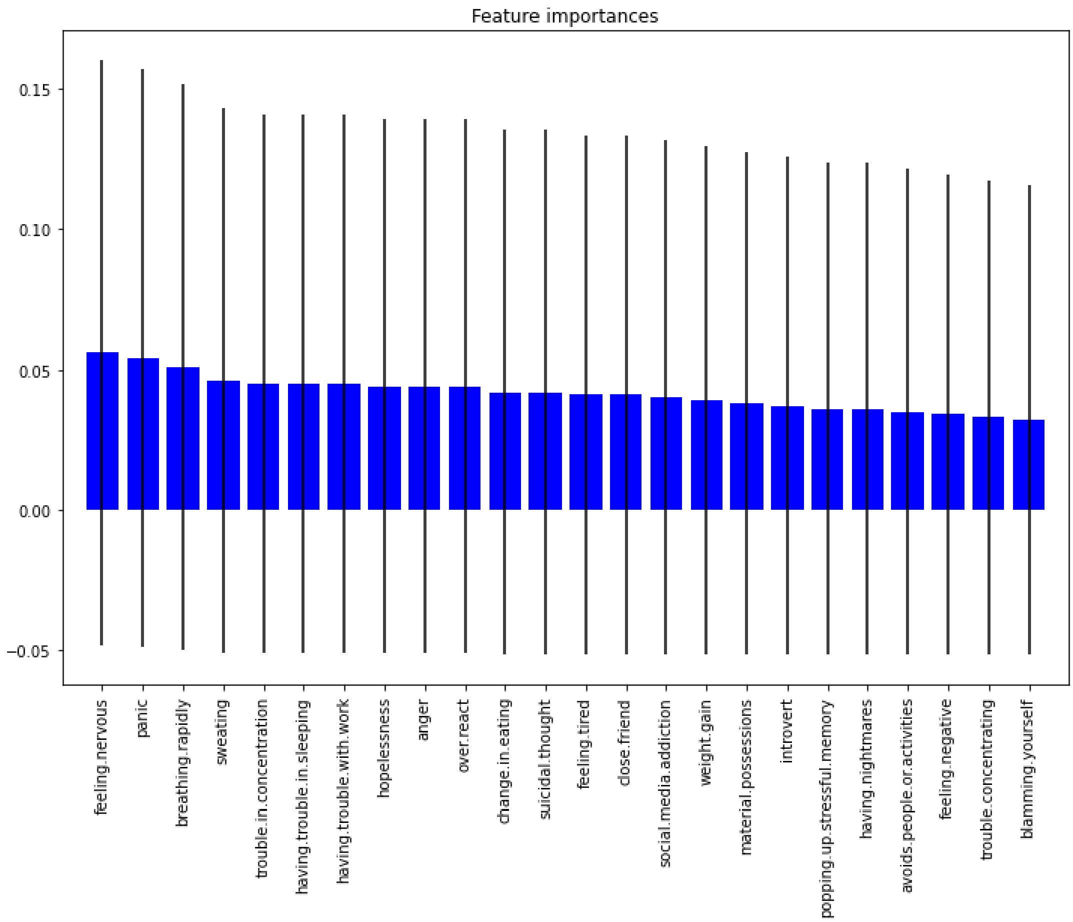


In [17]:
```
data.describe()
```

Out[17]:

	feeling.nervous	panic	breathing.rapidly	sweating	trouble.in.concentration	having.trouble.in.sleeping	having.trouble.with.work	hopelessness
count	40960.000000	40960.000000	40960.000000	40960.000000	40960.000000	40960.000000	40960.000000	40960.000000
mean	0.200000	0.200000	0.200000	0.200000	0.200000	0.200000	0.200000	0.200000
std	0.400005	0.400005	0.400005	0.400005	0.400005	0.400005	0.400005	0.400005
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 25 columns



In [18]:
```
########## Split Features and Target Varible ############
X = data.drop(columns='Disorder')
y = data['Disorder']
```

In [19]:
```
################# Splitting into Train -Test Data #######
from sklearn.model_selection import train_test_split

X_train, X_rem, y_train, y_rem = train_test_split(X,y, train_size=0.8, random_state =42)


# Now since we want the valid and test size to be equal (10% each of overall data).
# we have to define valid_size=0.5 (that is 50% of remaining data)

test_size = 0.5
X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem, test_size=0.5)


print(X_train.shape), print(y_train.shape)
print(X_valid.shape), print(y_valid.shape)
```

```
(32768, 24)
(32768,)
(4096, 24)
(4096,)
```

Decision Tree model

In [21]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,f1_score, precision_score, recall_score, classification_report

clf = DecisionTreeClassifier(max_depth=3,min_samples_leaf = 35)
clf.fit(X_train,y_train)
```

Out[21]:
```
DecisionTreeClassifier(max_depth=3, min_samples_leaf=35)
```

In [22]:
```
y_pred = clf.predict(X_test)
```

In [23]:
```
dc_accuracy = accuracy_score(y_pred, y_test)
dc_accuracy
```

Out[23]:
```
0.795166015625
```

In [24]:
```
print(classification_report(y_test, y_pred))
```

```
    accuracy                           0.80      4096
   macro avg       0.70      0.80      0.73      4096
weighted avg       0.69      0.80      0.73      4096
```

LogisticRegression Model

In [25]:
```python
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(penalty='l2',C=0.0000001, solver = "liblinear", max_iter=200, multi_class='ovr', tol=0.002)
logreg.fit(X_train, y_train)

y_pred_class = logreg.predict(X_test)

lr_accuracy = accuracy_score(y_test,y_pred_class)
print("Training Accuracy: ",logreg.score(X_train,y_train))
print("Test Accuracy: ", lr_accuracy)
```

```
Training Accuracy:  0.799652099609375
Test Accuracy:  0.798828125
```

SVM Model

In [26]:
```python
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
```

In [27]:
```python
svclassifier = SVC(kernel = 'linear', C = 0.00002)

svc_model=svclassifier.fit(X_train, y_train)
# predict the values
svc_pred = svclassifier.predict(X_test)

svc_accuracy = accuracy_score(y_test,svc_pred)

print("Accuracy: ",svc_accuracy)
```

```
Accuracy:  0.593994140625
```

XGBoost

In [28]:
```python
from xgboost import XGBClassifier
```

In [29]:
```python
xgb_model = XGBClassifier(learning_rate= 1000, max_depth=3, min_child_weight=5,
              n_estimators=5, n_jobs=-1, gamma=10)
xgb_model.fit(X_train, y_train)
```

```
/opt/anaconda3/lib/python3.8/site-packages/xgboost/sklearn.py:1146: UserWarning: The use of label encoder in XGBClassifier is d
eprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=F
alse when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num
_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

```
[16:27:58] WARNING: /Users/travis/build/dmlc/xgboost/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation met
ric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like
to restore the old behavior.
```

Out[29]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=10, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=1000, max_delta_step=0, max_depth=3,
              min_child_weight=5, missing=nan, monotone_constraints='()',
              n_estimators=5, n_jobs=-1, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=None, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [30]:
```python
xgb_pred  = xgb_model.predict(X_test)
```

In [31]:
```python
xgb_accuracy = accuracy_score(y_test, y_pred)
xgb_accuracy
```

Out[31]:
```
0.795166015625
```

FFNN (Feed Forward Neural Network)

In [32]:
```
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [33]:
```
# create a sequential model
model = Sequential()

# add the hidden layer
model.add(Dense(input_dim=24,units=5, activation='tanh'))

model.add(Dense(input_dim=12,units=5, activation='tanh'))

# add the output layer
model.add(Dense(input_dim=4,units=1,activation='sigmoid'))

# define our loss function and optimizer
model.compile(loss='binary_crossentropy',
              # Adam is a kind of gradient descent
              optimizer=optimizers.Adam(lr=0.01),
              metrics=['accuracy'])
```

```
/opt/anaconda3/lib/python3.8/site-packages/tensorflow/python/keras/optimizer_v2/optimizer_v2.py:374: UserWarning: The `lr` argu
ment is deprecated, use `learning_rate` instead.
  warnings.warn(
```

In [34]:
```
history = model.fit(X_train, y_train, batch_size = 32, epochs = 50, validation_data = (X_valid, y_valid))
history
```

```
Epoch 1/50
1024/1024 [==============================] - 3s 2ms/step - loss: -39.2989 - accuracy: 0.3979 - val_loss: -76.2462 - val_accur
acy: 0.4036
Epoch 2/50
1024/1024 [==============================] - 1s 1ms/step - loss: -112.6837 - accuracy: 0.3992 - val_loss: -149.0477 - val_acc
uracy: 0.4036
Epoch 3/50
1024/1024 [==============================] - 2s 2ms/step - loss: -185.4497 - accuracy: 0.3992 - val_loss: -221.7492 - val_acc
uracy: 0.4036
Epoch 4/50
1024/1024 [==============================] - 2s 2ms/step - loss: -258.1535 - accuracy: 0.3992 - val_loss: -294.3581 - val_acc
uracy: 0.4036
Epoch 5/50
1024/1024 [==============================] - 1s 1ms/step - loss: -330.7133 - accuracy: 0.3992 - val_loss: -366.8860 - val_acc
uracy: 0.4036
Epoch 6/50
1024/1024 [==============================] - 1s 1ms/step - loss: -403.2484 - accuracy: 0.3992 - val_loss: -439.4170 - val_acc
uracy: 0.4036
Epoch 7/50
1024/1024 [==============================] - 1s 1ms/step - loss: -475.9418 - accuracy: 0.3992 - val_loss: -512.1500 - val_acc
```

In [35]:
```
FFNN_train_accuracy = model.evaluate(X_train, y_train)[1]
print("Training Accuracy = %s" % FFNN_train_accuracy)
```

```
1024/1024 [==============================] - 1s 1ms/step - loss: -3634.8838 - accuracy: 0.3992
Training Accuracy = 0.399169921875
```

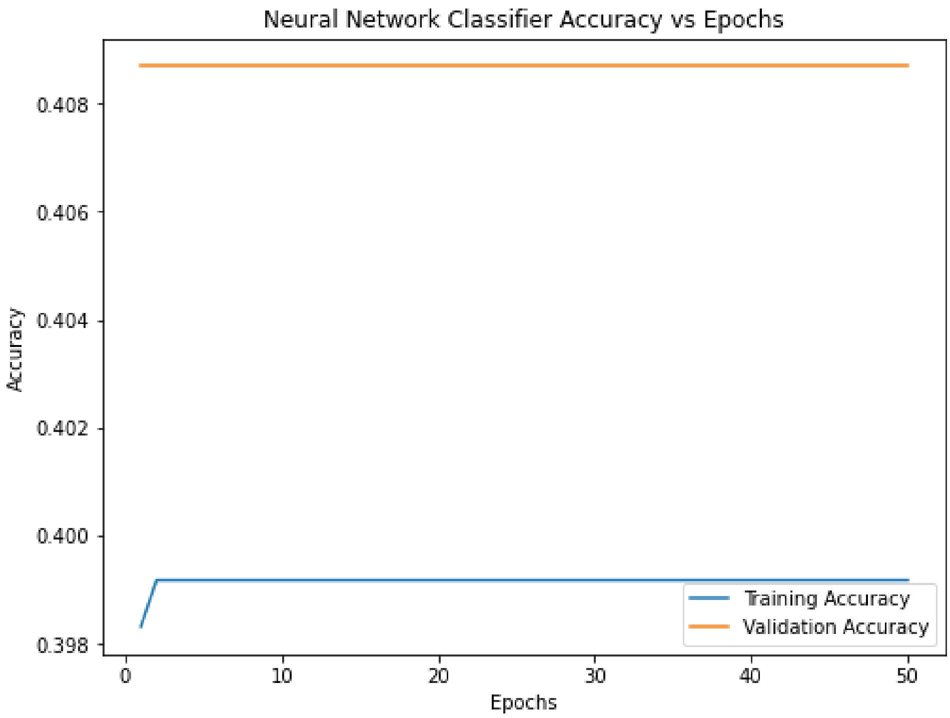
In [36]:
```
FFNN_test_accuracy = model.evaluate(X_test, y_test)[1]
print("Testing Accuracy = %s" % FFNN_test_accuracy)
```

```
128/128 [==============================] - 0s 1ms/step - loss: -3660.8799 - accuracy: 0.4031
Testing Accuracy = 0.403076171875
```

Accuracy VS Epochs Plot

In [37]:
```
epochs = range(1, 51)
train_accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]
```

In [38]:
```
plt.figure(figsize = (8,6))
plt.plot(epochs, train_accuracy, label="Training Accuracy")
plt.plot(epochs, val_accuracy, label="Validation Accuracy")
plt.title("Neural Network Classifier Accuracy vs Epochs")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
```



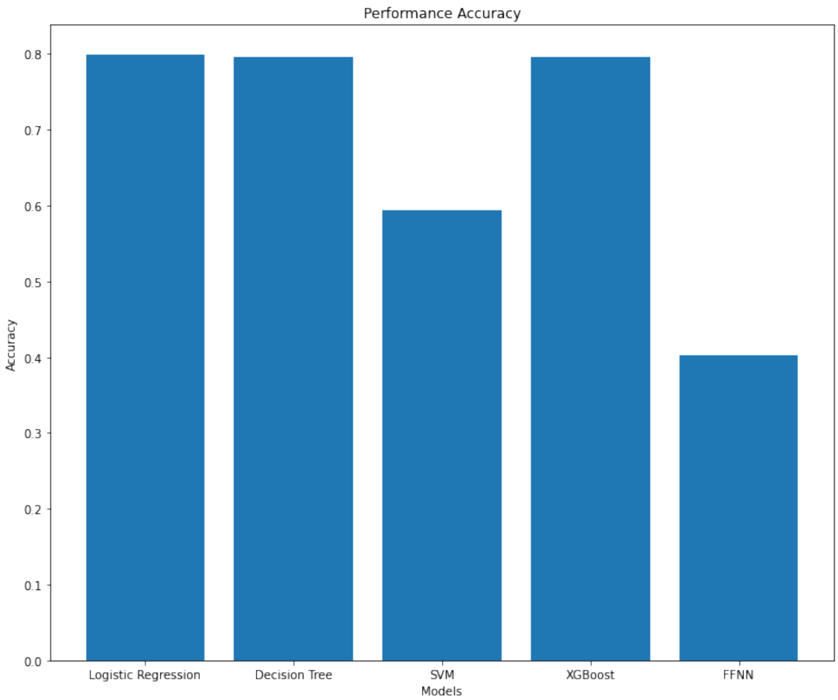
Comparision Between Accuracy Of Different Models

In [39]:
```
accuracy = [lr_accuracy, dc_accuracy, svc_accuracy, xgb_accuracy, FFNN_test_accuracy]
```

In [40]:
```
label = ["Logistic Regression", "Decision Tree", "SVM", "XGBoost", "FFNN"]
```

In [41]:
```
plt.figure(figsize = (12,10))
plt.bar(label, accuracy)
plt.title("Performance Accuracy")
plt.xlabel("Models")
plt.ylabel("Accuracy")
```

Out[41]:
```
Text(0, 0.5, 'Accuracy')
```


Performance Accuracy
Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Logistic Regression
Decision Tree
SVM
XGBoost
FFNN
Models